Università di Pisa
Scuola di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Laurea Magistrale

Implementazione su FPGA di un algoritmo per la stima dello Stato di Carica di batterie al litio-polimero

Candidato: Francesco Mengali

Relatore: Prof. Roberto Saletti .....

Relatore: Ing. Federico Baronti .....

Introduzione

Le batterie al litio sono ormai consolidate in applicazioni low-power consumer (come smart phones e laptops). Grazie all'elevata densità di potenza e di energia che sono in grado di fornire stanno diventando attraenti anche per applicazioni high-power come veicoli elettrici. Nonostante la chimica al litio sia una tecnologia ormai consolidata, essa presenta molti aspetti ancora in fase di ricerca, tra cui la scelta dei materiali più adatti per la realizzazione delle celle e lo sviluppo di algoritmi e circuiti elettronici per un efficiente utilizzo della batteria. Le batterie al litio devono infatti essere costantemente monitorate da un opportuno sistema di controllo, il Battery Management System (BMS), in quanto possono essere seriamente danneggiate qualora i valori di tensione o di temperatura eccedano i limiti di sicurezza.
Uno dei requisiti fondamentali del BMS è quello di fornire una stima accurata sullo stato di carica (State of Charge, SoC) e sullo stato di salute (State of Health, SoH) della batteria; grazie a tale stima gli algoritmi di gestione della batteria sono in grado di garantire un'elevata affidabilità ed efficienza. Una stima di questo tipo richiede un accurato modello che rappresenti con precisione il comportamento, sia statico che dinamico, della batteria. Lo sviluppo di tale modello, dovendo esso tenere conto di fattori variabili nel tempo e dipendenti dall'utilizzo, richiede una identificazione on-line dei parametri della batteria.
Lo scopo di questa tesi è quello di realizzare su field-programmable gate array (FPGA) un sistema per la stima dello stato di carica di una cella di batteria al litio. Gli FPGA sono infatti sempre più spesso utilizzati per applicazioni di tipo digital signal processing, soprattutto per applicazioni in cui è possibile realizzare una parallelizzazione del flusso di dati, in quanto consentono

di ottenere, oltre ad un'elevata flessibilità, anche ottime prestazioni. Inoltre, grazie alla possibilità offerta da moderni design tools di svolgere il progetto in un ambiente favorevole allo sviluppo di algoritmi, il flusso di progetto su FPGA risulta semplice ed efficace.
Dopo una breve presentazione dell'algoritmo di stima sarà esposta l'implementazione del sistema su FPGA e saranno mostrati i risultati sperimentali ottenuti da tests eseguiti su una cella di batteria agli ioni di litio di tipo Nickel-Manganese-Cobalto (NMC).

Indice

Capitolo 1

Sistema di monitoraggio delle batterie (BMS)

Dopo una breve presentazione delle tipologie di batterie al litio disponibili in commercio, in questo capitolo viene presentata la struttura ed il funzionamento di un generico BMS, evidenziandone alcuni aspetti relativi al suo processo di progettazione.

1.1 Le batterie al litio

Le batterie al litio sono diventate, negli ultimi anni, molto richieste per applicazioni high-power, come ad esempio nei veicoli elettrici o ibridi o nelle *smart grids* [2], grazie ad una tecnologia che consente di ottenere elevate densità di potenza e di energia, oltre ad una elevata efficienza di conversione.
Tra le tecnologie già applicabili ai veicoli elettrici, la più efficiente risulta essere quella delle batterie agli ioni di litio (Li-ion). Infatti, rispetto ad altri tipi di batterie quali batterie piombo-acido, batterie nickel-cadmio (Ni-Cd) o batterie nickel-metallo idruro (Ni-MH), le batterie agli ioni di litio risultano superiori in termini di densità di energia e potenza (specifica e volumetrica), caratteristiche che le rendono meno ingombranti e più leggere. Inoltre le batterie agli ioni di litio hanno un intervallo di temperature di funzionamento più ampio, velocità di ricarica più elevata, tempo di vita relativamente lungo e un basso tasso di auto-scarica, caratteristiche che le hanno rese molto dif-

fuse [3]. Tuttavia, a differenza delle batterie sopra elencate, le batterie agli ioni di litio possono essere seriamente danneggiate se i valori di tensione, corrente e temperatura superano i limiti di sicurezza; per questo motivo tali batterie devono sempre essere monitorate e gestite da un opportuno sistema di controllo, il Battery Management System (BMS).

Le batterie agli ioni di litio sono costituite da un catodo (di ossido di metallo), da un anodo (di carbonio poroso) e da un elettrolita come conduttore. Durante la scarica gli ioni si separano dall'anodo e si muovono verso il catodo attraverso l'elettrolita; contemporaneamente dall'anodo vengono rilasciati elettroni, i quali formano la corrente che scorre nel carico. Il processo di carica inverte la direzione di tale flusso consentendo agli ioni di tornare all'anodo.
Le batterie agli ioni di litio si distinguono per il materiale del catodo, che costituisce la maggiore fonte di ioni litio di tutta la batteria. I materiali prevalentemente utilizzati per la realizzazione dei catodi sono [3]:

1. **Ossido di Cobalto ($LiCoO_2$, LCO)**: esso fornisce alla batteria una capacità relativamente elevata (circa 155 mAh/g) ed un'alta tensione nominale (circa 3.9 V). Tuttavia la scarsa disponibilità di cobalto comporta costi elevati, per cui sono stati studiati nuovi materiali adeguati ad applicazioni di larga scala, come ad esempio i veicoli elettrici.
2. **Ossido di Manganese ($LiMn_2O_4$, LMO)**: il vantaggio principale del LMO è il costo relativamente basso del manganese, che è abbondante in natura e non è tossico per l'ambiente; inoltre esso garantisce alla batteria un lungo tempo di vita a temperatura ambiente. Tuttavia LMO ha una capacità più bassa ($100 \div 120$ mAh/g) e una più elevata velocità di perdita della capacità immagazzinata.
3. **Fosfato di Ferro ($LiFePO_4$, LFP)**: grazie alla presenza del fosfato nel materiale del catodo, le LFP sono più affidabili e sicure degli altri materiali per catodi, come LCO o LMO. I fosfati sono infatti estremamente stabili in caso di condizioni di sovraccarica o di corto-circuito (in tal caso non avviene la fuga termica) e sono in grado di sopportare un più ampio intervallo di temperature. Inoltre, le batterie LFP

anch'esse piuttosto economiche e consentono di ottenere celle con una capacità specifica di circa 160 mAh/g ed una tensione media di 3.40 V. Per tutte queste caratteristiche risultano quindi tra le più adatte per l'applicazione nel campo dei veicoli elettrici.

4. **Ossido di Nickel-Manganese-Cobalto ($\mathbf{LiNi_{1-y-z}Mn_yCO_zO_2}$, NMC)**: il materiale NMC ha una struttura simile a quella degli altri tipi con caratteristiche sostanzialmente equivalenti a quelle del LCO, ma, rispetto ad esse, ha come vantaggio un minor costo delle materie prime e una maggiore stabilità termica; esso ha inoltre un'ottima capacità specifica, pari a circa 200 mAh/g. Purtroppo, la loro scarsa stabilità termica, se confrontata con quella del LFP, comporta la necessità di un complesso e accurato sistema di monitoraggio in quanto un utilizzo improprio della batteria potrebbe degradarne le prestazioni.

	LCO	LMO	LFP	NMC
tensione nominale	3.60V	3.80V	3.30V	3.60/3.70V
massima tensione	4.20V	4.20V	3.60V	4.20V
cicli di scarica	500–1,000	500–1,000	1,000–2,000	1,000–2,000
energia specifica	150–190Wh/kg	100–135Wh/kg	90–120Wh/kg	140-180Wh/kg
sicurezza	scarsa	media	molto buona	buona
fuga termica	150°C	250°C	270°C	210°C
costo	alto	basso	medio	medio-alto

Figura 1.1: Tabella riassuntiva delle caratteristiche delle batterie agli ioni di litio

In conclusione i materiali di catodo più adatti ad applicazioni per veicoli elettrici sono il LFP e il NMC: il primo in quanto non solo garantisce una tensione nominale ed una capacità pienamente adeguata (sebbene non paricolarmente elevata), ma anche un ottimo livello di sicurezza e un tempo di vita di migliaia di cicli (pari a $1000 \div 2000$, superiore rispetto ai $500 \div 1000$ di LCO e LMO); il NMC in quanto garantisce un'ottima energia e potenza specifica e un tempo di vita pari a quello del LFP, a fronte di un minore

livello di sicurezza intrinseco, il quale può essere reso adeguato dal BMS.

Tra le batterie agli ioni di litio più diffuse vi sono le batterie al litio-polimero (lithium-ion polymer, o semplicemente Li-polymer). Tutte le batterie sopra elencate possono essere realizzate come batterie al litio-polimero, e per questo motivo la tipologia "litio-polimero" non è considerata come una chimica specifica per le batterie al litio. La differenza rispetto alle normali batterie agli ioni di litio consiste nel fatto che il tradizionale separatore poroso è sostituito da un elettrolita solido di tipo polimerico.
Le celle al litio-polimero sono disponibili in case di lamina di tipo flessibile, simili a sacchetti. Mentre una tradizionale cella agli ioni di litio ha bisogno di una custodia rigida per mantenere premuti gli elettrodi, la cella al litio-polimero utilizza fogli laminati che non hanno bisogno di compressione; per questo motivo le celle al litio-polimero risultano più leggere e possono essere progettate di forma qualsiasi, garantendo quindi un'ottima flessibilità in fase di progetto.

1.2 Caratteristiche e funzioni di un BMS

Le batterie al litio, come evidenziato nel paragrafo precedente, costituiscono una scelta promettente come mezzo di immagazzinamento dell'energia nei veicoli elettrici. Esse sono costituite da decine di celle ad elevata capacità connesse in serie in modo da ottenere valori di tensione di centinaia di volt, come tipicamente richiesto da applicazioni high-power. In queste applicazioni le batterie al litio devono lavorare entro la zona di funzionamento sicuro ed affidabile, in quanto, operando al di fuori di tale zona, le prestazioni si degraderebbero rapidamente e si avrebbero problemi di sicurezza (possibilità che avvengano reazioni di combustione tra gas prodotti dalla batteria o possibilità di fuga termica). Ciò restringe gli intervalli di tensioni e temperature nei quali possono operare. Dunque problemi quali sicurezza, affidabilità e costi, impongono dei limiti ad un largo impiego di tali batterie.
Inoltre, a causa della loro struttura (connessione serie) e della differenza di capacità inevitabilmente presente tra le celle, si verifica una progressiva degradazione della distribuzione della carica tra le varie celle. Anche se si

ipotizza che tutte le celle abbiano la stessa capacità (le differenze sono dovute a errori di matching, quindi al massimo di pochi punti percentuali), lo sbilanciamento è comunque presente a causa della diversa velocità di autoscarica e della diversa temperatura di lavoro delle celle.
Questo fenomeno di sbilanciamento può portare ad avere una cella della batteria completamente carica (o scarica) prima delle altre; se ciò accade, non può più essere fornita (o prelevata) energia alla batteria per evitare sovraccarica (o scarica profonda) della cella che ha raggiunto il limite per evitare che essa si danneggi. Lo sbilanciamento, quindi, costituisce una delle principali cause di degrado delle prestazioni di una batteria.
Assieme alla batteria è dunque sempre presente un opportuno sistema di gestione, il BMS, che mantiene lo stato di ciascuna cella della batteria entro i limiti di (buon) funzionamento e che si occupa di effettuare un bilanciamento della carica tra le celle, in modo tale da evitare danneggiamenti dovuti ad una distribuzione di carica non adeguata. Oltre a questa funzione di protezione, il BMS deve essere in grado di stimare lo stato della batteria, in modo che si possa predire la quantità di energia che può ancora essere fornita al carico e quindi predire la durata della batteria, nonché il suo tempo di vita.

Le variabili di stato principali di una batteria sono lo stato di carica (State of Charge, SoC), e lo stato di salute (State of Health, SoH).

- Il SoC indica la carica residua della batteria ed è espresso come percentuale della capacità nominale [4]. La sua stima consente di mantenere la batteria nei limiti ottimali di funzionamento e di valutare quanto tempo rimane prima della scarica completa.
- Il SoH è un parametro importante per la stima del tempo di vita, in quanto riflette le condizioni generali della batteria; tali condizioni variano continuamente nel tempo come conseguenza del progressivo degrado della capacità indotto dalla variazione nel tempo del valore della capacità e della resistenza rispetto al valore nominale [5].

Solitamente il SoC è fornito come rapporto della carica residua rispetto alla capacità effettiva della cella; per questo motivo è indispensabile la conoscenza

del SoH, la quale consente di risalire all'effettiva carica immagazzinata nella cella. La stima di tali variabili è dunque essenziale per il BMS, in quanto grazie a tale stima esso è in grado di prolungare (il più possibile) la vita del pacco batteria e di mantenerlo in una condizione di funzionamento ottimale per poter soddisfare le esigenze specifiche dell'applicazione.
Per raggiungere questi obiettivi, assieme all'obiettivo già discusso di protezione dal danneggiamento, è necessario che un BMS includa le seguenti funzionalità [6, 7]:

1. **misura dei parametri**, ovvero la tensione dell'intero pacco, la tensione di ogni cella, la carica totale, la carica di ogni cella, la corrente totale, la temperatura e altri parametri di sicurezza quali isolamento, impedenza, presenza di fumo etc. L'accuratezza delle misure, in particolare di tensione e corrente, dipende dal tipo di batteria; le batterie Li-$FePO_4$ in particolare necessitano di una misura accurata della tensione, in quanto la caratteristica SoC-tensione a circuito aperto (Open Circuit Voltage, OCV), utilizzata per la stima del SoC (cfr. capitolo 2), è molto piatta (quasi orizzontale) nell'intervallo 20-80% del SoC, che è il tipico intervallo di funzionamento della batteria. Sono necessarie dunque accuratezze dell'ordine di 1-2 mV sulla misura della tensione.
2. **stima delle variabili di stato**, ovvero di SoC e SoH (descritta nel capitolo 2).
3. **bilanciamento della carica**, con una delle varie tecniche di bilanciamento descritte in seguito, che dipendono dalla tipologia di batteria e di BMS.
4. **controllo della carica**. E' necessario l'utilizzo di un opportuno sistema di controllo in fase di carica della batteria per diversi motivi. Solitamente la velocità di carica è un parametro di qualità richiesto dell'utente, il quale ha come obiettivo quello di ottenere una ricarica nel minore tempo possibile. Tuttavia ci sono dei limiti, dovuti alla chimica o alla struttura della batteria, sulla massima correte di carica; tali limiti devono essere imposti dal BMS. Inoltre, può darsi che una carica veloce della batteria non sia anche efficiente in termini di quantità di carica totale ceduta alla batteria. Il BMS dunque deve gestire,

ottimizzare e proteggere il processo di carica.

5. **diagnosi di malfunzionamenti**, che riguardano non solo la batteria ma anche l'elettronica di controllo.
6. **sicurezza**. È indispensabile un continuo monitoraggio della tensione e della temparatura di cella. In caso avvenga un guasto o un malfunzionamento, il BMS deve attivare il sistema di allarme e gestirlo in modo tale che il problema sia risolto senza danni.
7. **comunicazione** con l'utente o con il sistema di controllo del veicolo. Solitamente viene utilizzata una comunicazione CAN.

1.3 Architettura di un BMS

La scelta dell'architettura di un BMS si basa spesso sulla struttura fisica della batteria, che dipende dalla specifica applicazione. Nel caso di applicazioni high-power, dove sono presenti anche più di un centinaio di celle connesse in serie per ottenere il livello di tensione richiesto, la batteria risulta costituita da moduli (contenenti più celle connesse in serie) uniti insieme per formare un pacco. Dunque la batteria può essere vista come costituita da più livelli, ciascuno dei quali necessita di un opportuno controllo da parte del BMS.

Tra le varie soluzioni possibili, un BMS costituito da una struttura gerarchica rappresenta una delle soluzioni più flessibili ed efficienti, in quanto suddivide le funzioni da svolgere distribuendole sui vari livelli gerarchici, e ciò rende la progettazione di ciascun livello semplice e rapida; inoltre, in questo modo le funzioni più critiche di monitoraggio possono essere implementate su più livelli, incrementando l'affidabilità del sistema [8]. Una architettura di questo tipo, dunque, costituisce la soluzione più generale che si possa pensare.
Questa struttura prevede la presenza di una unità di controllo per ciascuna cella (Cell Monitor Unit, CMU) per il livello inferiore, una unità di controllo di ciascun modulo (Module Monitor Unit, MMU) per il livello intermedio, e una unità di controllo del pacco (Pack Management Unit, PMU) per il livello superiore che gestisce l'intera batteria. In particolare, la PMU deve stimare il SoC e il SoH per ciascuna cella, in modo da poter calcolare la quantità

di energia attualmente presente nella batteria per gestire il flusso di potenza da o verso il carico [8]. Per realizzare la comunicazione tra PMU e MMU si utilizza solitamente un Controller Area Network (CAN) bus isolato galvanicamente.

È presente inoltre un insieme di interruttori la cui funzione è quella di proteg-

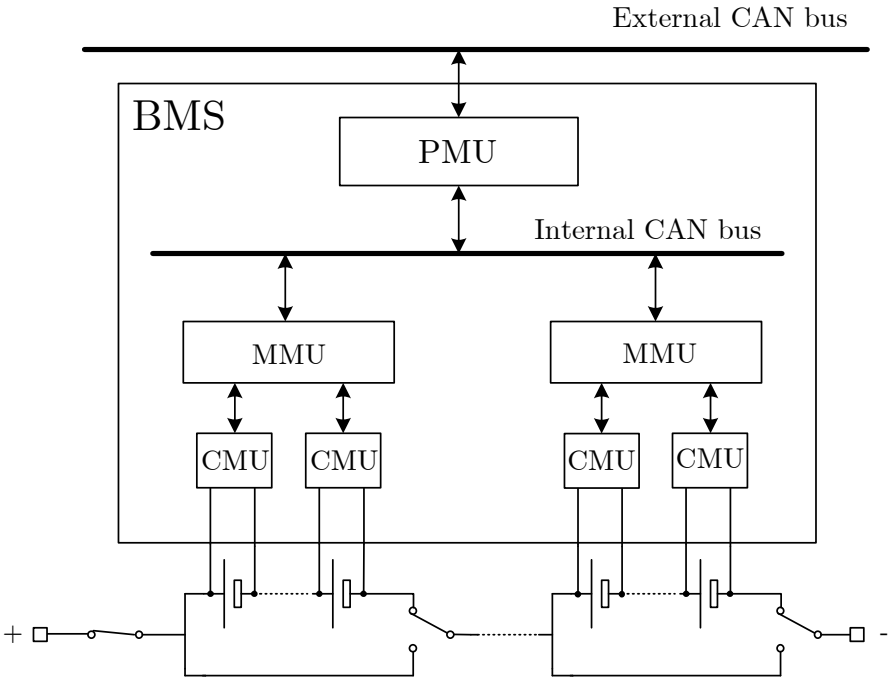

Figura 1.2: Architettura gerarchica di un BMS

gere la batteria dalla sovra-carica, dalla scarica profonda e da effetti dovuti ad un eccesso dai limiti di sicurezza, il Main Switch Unit (MSU). Poiché tale unità è in grado di escludere un segmento della stringa dal percorso della corrente, oltre ad una funzione di protezione esso può svolgere un'altra fondamentale funzione, ovvero l'*equalizzazione della carica*. Se un segmento della batteria raggiunge lo stato di carica completa, esso può essere disconnesso dagli altri segmenti della stringa in modo tale che essi possano essere caricati senza che avvenga alcun danneggiamento. A seconda del tipo di batteria, questa operazione di equalizzazione può essere realizzata anche a livello di cella [9].

Molte tecniche di equalizzazione di carica sono presenti in letteratura. Sostanzialmente si suddividono in metodi di equalizzazione passivi e attivi. Quelli passivi consistono nell'utilizzo di *bleeding resistors* che hanno la funzione di scaricare la carica in eccesso della cella più carica. Quelli attivi invece prevedono un trasferimento di carica dalla cella o dall'intero modulo più carico verso la cella o il modulo meno carico, con diverse modalità. Una buona panoramica dei metodi di equalizzazione della carica è raccolta in [10].
Nonostante questa seconda soluzione possa sembrare la migliore, è importante sottolineare che deve essere raggiunto un buon compromesso tra la complessità circuitale necessaria per realizzare una equalizzazione attiva e l'efficienza effettivamente ottenibile; tale compromesso non è detto che sia più competitivo rispetto ad una semplice equalizzazione passiva.

Capitolo 2

Stima dello stato di carica (SoC) e dello stato di salute (SoH)

Come visto nel precedente capitolo, lo stato di carica è una variabile essenziale per la gestione, da parte del BMS, della batteria. In questo capitolo viene presentata la definizione di SoC e vengono riportati i più comuni metodi di stima dello stato di carica, nonché, nella parte conclusiva del capitolo, dello stato di salute.

2.1 Introduzione al SoC

Come anticipato nel precedente capitolo, il SoC è un parametro che misura la carica contenuta nella batteria calcolata in termini di concentrazione di litio. Infatti durante la scarica il litio passa dall'anodo al catodo e la sua concentrazione agli elettrodi varia, dunque la quantità di litio contenuta negli elettrodi è legata alla quntità di carica attualmente disponibile nella cella. Per cui il SoC può essere utilizzato come indicatore dell'energia totale estraibile dalla cella ad un certo istante ed in determinate condizioni di utilizzo.
La conoscenza del SoC consente quindi di mantenere la batteria nei limiti di funzionamento sicuro ed efficiente, di valutare per quanto tempo ancora la batteria può erogare corrente prima di scaricarsi e di effettuare un efficace bilanciamento della celle. Dunque una conoscenza accurata del SoC è indispensabile per garantire l'efficienza, la sicurezza e l'affidabilità del sistema.

Poiché il SoC è una grandezza non direttamente osservabile, in quanto la carica immagazzinata nella batteria può essere misurata solo quando viene estratta, la conoscenza del SoC avviene attraverso un processo di stima la cui complessità varia a seconda dell'applicazione. Per alcune applicazioni, tipicamente quelle che riguardano sistemi low power, i metodi di stima sono semplici in quanto una conoscenza approssimativa del SoC non influenza più di tanto le prestazioni del sistema e per tali sistemi i requisiti di sicurezza sono tipicamente meno stringenti. In altre applicazioni tuttavia, come nel caso in questione di batterie high power agli ioni di litio, l'esigenza di prestazioni elevate ed i requisiti di sicurezza impongono l'utilizzo di tecniche di stima avanzate, alcune delle quali (le più comuni) sono descritte nel presente capitolo.

2.2 Definizione di SoC

A causa della natura non direttamente osservabile del SoC, la sua definizione non è univoca, e può essere espressa in diversi modi. In [4, 5, 11] il SoC è definito come il rapporto (espresso in percentuale) tra la carica rimanente nella cella (Q_c) e la capacità nominale della cella (Q_n), dove la carica rimanente è la carica residua che può essere estratta dalla cella, e la capacità nominale è la massima quantità di carica immagazzinabile nella batteria (fornita dal costruttore). Si ha quindi

$$\mathrm{SoC} = \frac{Q_c}{Q_n}$$

Con una definizione di questo tipo, però, lo stato di carica della batteria non è normalizzato alla carica totale che effettivamente è immagazzinabile nella batteria in quanto questa può differire dalla capacità nominale, ed inoltre dipende dall'invecchiamento della batteria. Ciò significa che quando la batteria è completamente carica, il SoC non è pari al 100%. Un'alternativa potrebbe quindi essere quella di normalizzare il SoC con la carica totale effettivamente estraibile dalla cella, ma poiché essa cambia nel tempo, tale normalizzazione deve essere aggiornata durante la vita della batteria. Quest'ultima definizione è quella adottata in questo lavoro di tesi.

Una definizione operativa del SoC è la seguente [12, 6]:

$$\mathrm{SoC}(t) = \mathrm{SoC}(t_0) + \frac{1}{Q_n} \int_{t_0}^{t} \eta_i \, i_L(\tau) \, \mathrm{d}\tau \tag{2.1}$$

dove $\mathrm{SoC}(t_0)$ è il SoC iniziale, Q_n è la capacità nominale, i_L è la corrente di carico, η_i è la *coulombic efficiency*, un parametro che tiene conto della diversa efficienza di trasferimento della carica tra batteria e carico nelle fasi di carica e scarica ($\eta_i = 1$ per la scarica, $\eta_i \leq 1$ per la carica). Solitamente nel caso di batterie al litio-polimero questo coefficiente viene trascurato in quanto molto prossimo a 1.

2.3 Metodi per la stima del SoC

2.3.1 Discharge test

Il "discharge test" è il metodo più affidabile per il calcolo del SoC in quanto consiste in una scarica controllata della batteria. Tale test comporta tuttavia la perdita dell'energia immagazzinata nella batteria e richiede l'attesa di un tempo necessario sia alla scarica, sia alla ricarica della batteria, il che rende tale metodo non adatto ad applicazioni in cui è richiesta una stima in tempo reale. Inoltre durante il test la batteria non è collegata al carico, quindi la funzionalità del sistema è interrotta.

2.3.2 Coulomb counting

E' una delle tecniche più semplici e più utilizzate che si basa sull'integrazione della corrente nel tempo per ottenere il valore della carica estratta o inserita nella batteria. Effettuando un rapporto tra questo valore e un riferimento di capacità della cella dipendente dal tipo di definizione di SoC scelto, si ottiene il valore della variazione di SoC tra il tempo t e l'istante t_0 in cuiè iniziata l'integrazione della corrente. Dunque il SoC stimato con il metodo di coulomb counting si ricava dall'equazione (2.1).
I risultati che si ottengono dall'impiego di questo metodo sono piuttosto precisi per un certo intervallo di tempo (che dipende dal periodo di cam-

pionamento del sensore e dalla frequenza di funzionamento del sistema) se SoC(t_0) è preciso. Tuttavia ci sono degli svantaggi, ovvero:

1. la determinazione di SoC(t_0) non è ricavabile dal semplice processo di integrazione della corrente, per cui occorre effettuare una stima. Tale stima però non è accurata: essa può ad esempio essere ottenuta tramite una look-up table basata sulla tensione a circuito aperto (Open-Circuit Voltage, OCV), ma il SoC così ottenuto non è accurato.
2. la coulombic efficiency (η_i) risulta fortemente influenzata dallo stato della batteria (ovvero dal valore di SoC, temperatura, corrente di carico, etc.) che è difficile da misurare, per cui tale dipendenza si traduce in un errore sul SoC stimato.
3. la precisione del sensore di corrente ed in particolare la presenza di un offset nella misura della corrente si traducono in un errore sul SoC. Poiché questa tecnica di integrazione avviene ad anello aperto, gli errori costanti nel tempo, come ad esempio un offset sulla corrente, tendono ad accumularsi e, nel tempo, a far divergere il valore dell'integrale.

A causa dei problemi sopra citati, la tecnica del Coulomb Counting richiede di effettuare periodicamente una calibrazione o una compensazione. Tale calibrazione però deve essere fatta in uno stato della batteria che sia noto, come ad esempio lo stato di carica completa. Tuttavia in molte applicazioni, come ad esempio nel caso di veicoli ibridi, raramente la batteria raggiunge la carica completa, per cui questa tecnica non può essere impiegata se non utilizzando un sistema di retro-azione che compensi gli errori.

2.3.3 Open circuit voltage method

In molte applicazioni, soprattutto per batterie che funzionano con basse correnti di carico, il SoC è ricavato a partire da una misura della tensione a circuito aperto OCV. Il SoC è infatti legato alla quantità di ioni di litio presenti nella cella, che ne determinano il valore di tensione; esiste dunque una corrispondenza biunivoca tra OCV e SoC, la cui conoscenza, noto il valore di OCV, consente di ricavare direttamente il SoC.
Purtroppo la misura dell'OCV richiede l'attesa di un tempo di rilassamento che solitamente è dell'ordine di alcune ore; durante questo tempo la batteria

deve essere scollegata dal carico e quindi il sistema resta inattivo. Esistono tuttavia diversi metodi di stima dell'OCV a tempo di esecuzione, che generalmente consistono in una modellizzazione dell'OCV o di parametri ausiliari inseriti all'interno di un appropriato modello della cella [13].

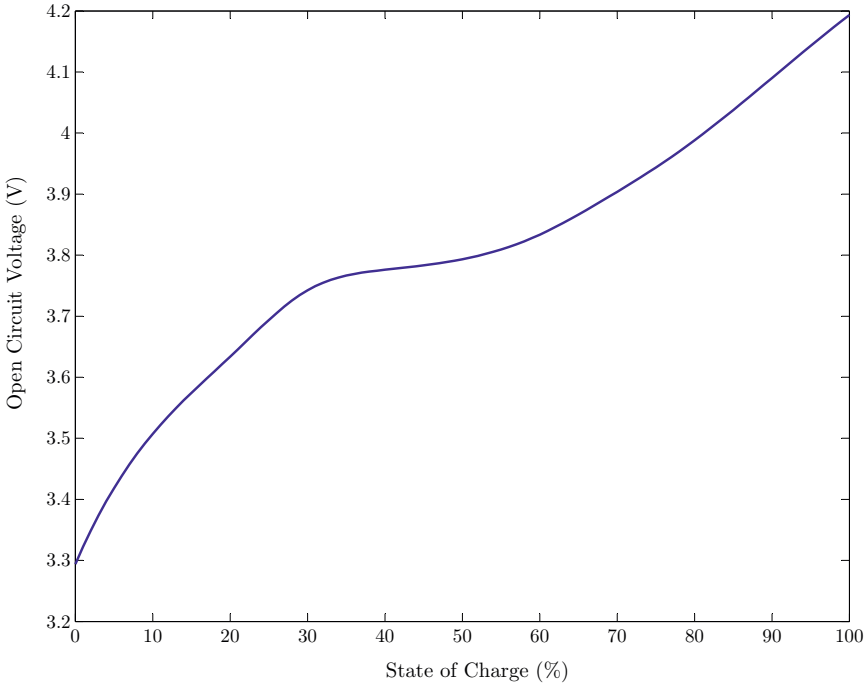

Figura 2.1: Caratteristica SoC-OCV per una batteria NMC

Come mostrato in Figura 2.1, la relazione statica tra OCV e SoC è intrinsecamente non lineare. Un errore sulla misura dell'OCV dunque si traduce in un errore sulla stima del SoC che dipende dal valore del SoC stesso. Per alcune batterie, in particolare le LFP, la caratteristica risulta piatta per un ampio intervallo di valori del SoC (in pochi mV il valore del SoC varia di varie decine di punti percentuali) e quindi è richiesta una misura dell'OCV molto accurata. Inoltre la curva SoC-OCV presenta una dipendenza (anche se debole) dalla temperatura ed un fenomeno di isteresi (in particolare nelle batterie LFP; nelle NMC questo fenomeno è molto meno marcato), che complicano il calcolo del SoC.

2.3.4 Model Based

Per le applicazioni che richiedono una stima on-line del valore del SoC un metodo efficace è quello che si basa su un modello equivalente della cella, il quale, simulando il funzionamento della cella, consente di ricavare le variabili di interesse (tra cui il SoC). Ovviamente, questo approccio è tanto più efficace quanto più il modello della cella è accurato; tuttavia, un modello troppo complesso può comportare la necessità di grandi risorse di calcolo e di lunghi tempi di calcolo, rendendo tale metodo poco adatto a molte applicazioni. E' dunque necessario trovar un compromesso tra complessità del modello e risorse / tempi di calcolo.

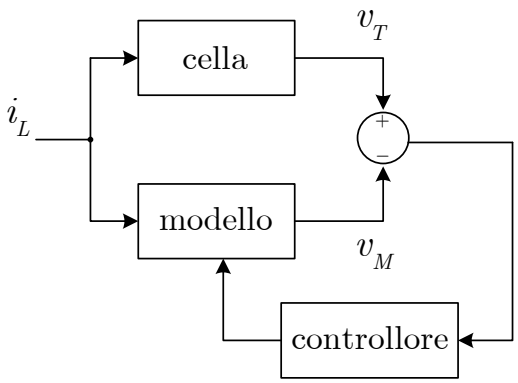

Figura 2.2: Schema di principio del metodo di stima basato sul modello di cella

Esistono vari metodi per la modellizzazione della cella di una batteria, generalmente suddivisi in [9, 17]:

1. **modelli elettro-chimici** (o *white-box*), i quali descrivono il comportamento della cella attraverso equazioni matematiche derivate dalle caratteristiche fisiche e chimiche dell'oggetto. Essi sono complessi e comportano costi in termini di risorse e tempi di calcolo, per cui risultano inadatti alla maggior parte delle applicazioni.
2. **modelli matematici** (o *black-box*), in cui le caratteristiche tensione-corrente misurate all'esterno sono utilizzate per determinare i coefficienti di funzioni matematiche e/o statistiche che descrivono il sistema.

3. **modelli elettrici** (o *gray-box*), che hanno sia una componente matematica, sia una corrispondenza fisica (elettrica); rispetto ai primi due consentono di ottenere tempi di calcolo più veloci e una maggiore semplicità implementativa anche se l'accuratezza del modello è spesso inferiore.

I metodi basati su modello di cella più spesso utilizzati sono quelli che impiegano modelli elettrici. Di essi fanno parte:

- modello circuitale
- filtri di Kalmann
- algoritmo misto

modello circuitale Questo metodo di stima si basa sull'impiego di un modello circuitale di cella come quello riportato in Figura 2.3.
Tale modello è costituito da due parti: la prima (a sinistra) rappresenta la

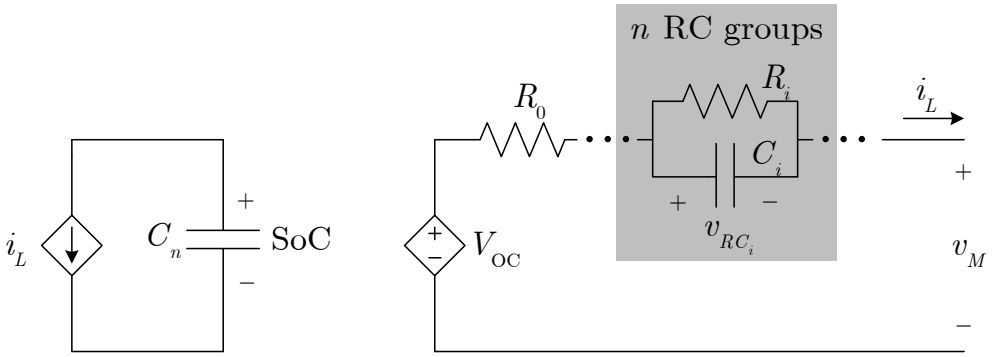

Figura 2.3: Circuito elettrico equivalente

carica della batteria come immagazzinata in un condensatore C_n, il cui valore è pari al valore della carica nominale Q_n diviso 1 V; la seconda (a destra) rappresenta le componenti della tensione v_T misurate ai capi della cella. Tale tensione è costituita da 3 componenti, ovvero la tensione a circuito aperto V_{OC}, la caduta sulla resistenza interna R_0 e la caduta di tensione su uno o più gruppi RC che tiene conto degli effetti di rilassamento:

$$v_T = V_{\mathrm{OC}} - V_{R_0} - \sum_i v_{RC_i} \tag{2.2}$$

Il generatore V_{OC} è controllato dal valore del SoC. Dunque noto v_T (tensione di cella, misurata) e noto il valore di i_L (corrente di carico, anch'essa misurata), si può risalire al valore di V_{OC} e, tramite la caratteristica SoC-OCV, al valore del SoC. Tutto ciò senza effettuare una misura diretta dell'OCV la quale, come visto nel precedente paragrafo, comporterebbe una perdita in termini di tempo di esecuzione.

Filtri di Kalman Questo metodo è diffuso in molte applicazioni in quanto consente una stima in tempo reale dello stato di un sistema dinamico [14, 15]. Il filtro di Kalman elabora in maniera ricorsiva insiemi di dati rumorosi per calcolare una stima (statistica) dello stato del sistema. In una prima fase esso produce una stima delle variabili di stato attuali e delle loro incertezze; successivamente queste stime sono aggiornate tramite una media pesata (maggior peso è dato alle grandezze con minore incertezza). Grazie alla natura ricorsiva dell'algoritmo, esso opera in tempo reale durante il funzionamento del sistema utilizzando soltanto i campioni attuali dei segnali di ingresso e i valori dei parametri (assieme alle loro incertezze) che descrivono lo stato all'istante precedente.
L'applicazione del metodo di stima basato sui filtri di Kalman in un BMS si basa sulla modellizzazione della cella in una forma tempo-discreta. Solitamente il modello in questione è costituito da un set di equazioni nel dominio tempo-discreto le cui variabili sono lo stato del sistema, l'insieme degli ingressi all'istante attuale e il rumore di processo che condiziona lo stato del sistema.
Vi sono vari modelli che possono essere impiegati per questo metodo di stima del SoC; i modelli più semplici, che tengono conto di pochi fattori, forniscono un stima poco accurata, ma aumentandone la complessità (aggiungendo ad esempio la dipendenza dalla temperatura, la presenza dell'isteresi etc.) si possono ottenere stime sempre più accurate al costo di una complessità crescente.

Algoritmo misto Per i metodi prcedentemente descritti sono stati enunciati i vantaggi/svantaggi che li caratterizzano. Unendo opportunamente due o più metodi si possono ottenere i vantaggi di entrambi.
Si consideri ad esempio la stima del SoC per un veicolo elettrico (Electric

Vehicle, EV). In tale applicazione si hanno condizioni di funzionamento relativamente semplici: quando il veicolo è in movimento le batterie sono prevalentemente in scarica (ad eccezione dei momenti in cui avvengono le frenate, in cui l'energia della frenata viene convertita e impiegata per ricaricare temporaneamente la batteria), inoltre spesso la batteria viene ricaricata completamente, portandosi quindi in uno stato noto. Per una applicazione di questo tipo si può utilizzare il metodo del Coulomb Counting, particolarmente adatto per la sua capacità di catturare anche variazioni rapide del SoC, assieme ad una correzione del SoC che può essere fatta in maniera esatta quando la batteria è completamente carica, o in maniera approssimata quando il veicolo è spento in quanto in tale condizione è possibile effettuare una misura dell'OCV e da essa risalire al SoC tramite la caratteristica SoC-OCV.
Un metodo di questo tipo, però, non è applicabile nei BMS per veicoli ibridi (Hybrid Electric Vehicles, HEV) in quanto la batteria non raggiunge mai lo stato di caria completa: quando il veicolo è in movimento la batteria subisce periodicamente cicli di carica e scarica in modo da contenere le variazioni di SoC all'interno di un intervallo ristretto. In questo caso il metodo di correzione del Coulomb Counting tramite la misura dell'OCV non è sufficiente.
Un buon metodo, che è stato impiegato in questa tesi e che verrà esposto nel dettaglio più avanti, è quello di effettuare una correzione dell'errore di integrazione del Coulomb Counting tramite l'impiego di un modello circuitale della cella dal quale ricavare una stima del comportamento dinamico della cella da utilizzare in un loop di retro-azione per correggere gli errori di integrazione.

2.4 Definizione di SoH

Lo stato di salute (SoH) è un parametro che riflette la condizione generale di una batteria. A causa dell'invecchiamento, infatti, il valore di capacità della cella e quindi dell'energia in essa immagazzinata si degrada. Il grado di perdita della capacità in funzione del tempo costituisce quindi una buona indicazione sullo stato di salute della cella; per questo motivo il SoH è definito

come:

$$\mathrm{SoH}(t) = \frac{Q_{\max}(t)}{Q_n} \tag{2.3}$$

dove $Q_{\max}(t)$ è la massima carica estraibile dalla cella all'istante t, mentre Q_n è la carica nominale.

2.5 Metodi di stima del SoH

La stima del SoH è piuttosto complessa, a causa del fatto che ci sono molti parametri coinvolti in tale stima. In particolare il SoH dipende dall'applicazione e dalle modalità di utilizzo della batteria: numero di cicli di scarica effettuati, tempo in cui la batteria resta inattiva (quindi avviene l'autoscarica), temperatura di riposo e temperatura di funzionamento etc., sono tutti parametri che condizionano il valore di $Q_{\max}(t)$.
Dal punto di vista dei parametri circuitali equivalenti della batteria, con l'invecchiamento avviene un aumento della resistenza interna e una diminuzione della capacità, che appunto si traducono in una riduzione della carica massima.
I metodi di stima del SoH sono sostanzialmente due [6], entrambi basati sull'impiego di un modello di invecchiamento.

Un primo metodo è un metodo ad anello aperto che prevede direttamente la perdita di capacità totale e le variazioni di resistenza interna grazie all'impiego di un modello di invecchiamento della cella. Tale modello può essere un modello chimico-fisico che stima lo stato delle reazioni interne basandosi su parametri come la resistenza dello strato di separazione, concentrazione degli ioni o altre quantità chimiche, oppure un modello matematico o circuitale empirico che si basa sulla misura dei parametri esterni della batteria (carica, corrente, tensione etc.).

Un secondo metodo è un metodo ad anello chiuso basato su un modello della batteria, analogo a quelli utilizzati nella stima del SoC, che utilizza tecniche avanzate di stima come filtri di Kalman o il metodo di stima nel

senso dei minimi quadrati (Least Square Method, LSM) per identificare i parametri di tale modello durante la vita della batteria e ottenere così una stima del SoH. È dunque un metodo simile all'algoritmo misto descritto in precedenza.
È importante sottolineare il fatto che una stima del SoH di questo tipo può tenere conto di modifiche del SoH dovute non solo all'invecchiamento ma anche ad altri fattori, come ad esempio elevate temperature di funzionamento, urti meccanici, corto-circuiti etc.

Capitolo 3

Algoritmo di stima realizzato

Viene adesso esposto l'algoritmo utilizzato in questa tesi per la stima del SoC. Si tratta di un algoritmo che utilizza parallelamente sia il metodo del Coulomb Counting, sia un modello circuitale della cella. Dato che la stima del SoC risulta tanto più accurata quanto più il modello della cella è preciso, ovvero quanto più esso riesce a riprodurre il comportamento reale della cella, l'algoritmo in questione comprende una stima in tempo reale (on-line) dei parametri della cella. Tali parametri, infatti, non sono costanti nel tempo, ma variano a seconda delle variazioni del SoC e della temperatura; un modello accurato, quindi, deve poter riprodurre queste variazioni.
In questo capitolo sono quindi presentati il modello della cella, gli algoritmi di stima del SoC e di identificazione dei parametri; è inoltre esposta l'analisi della stabilità del sistema e il suo comportamento in presenza di errori, studio effettuato da Sandro Rosi [16].
Considerato che il sistema di identificazione on-line dei parametri da noi utilizzato richiede la risoluzione di sistemi di equazioni nel senso dei minimi quadrati, nella parte conclusiva del capitolo è presentata una panoramica sugli algoritmi di risoluzione di tali problemi, con particolare riferimento alla loro implementazione su FPGA.

3.1 Struttura generale del sistema

La struttura complessiva del sistema di stima è riportata in Figura 3.1.
Esso è suddiviso in due parti:

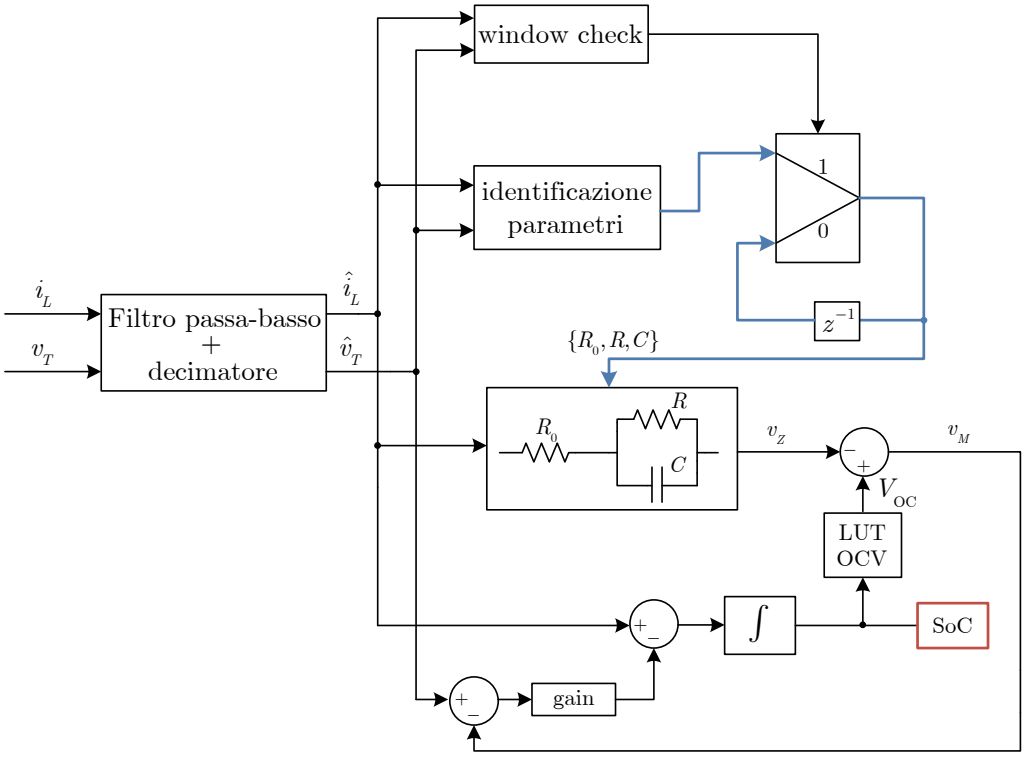

Figura 3.1: Schema a blocchi dell'algoritmo

1. Stima dello stato di carica
2. Identificazione dei parametri

La parte di stima del SoC, come accennato in precedenza, è realizzata tramite l'utilizzo un modulo di Coulomb-Counting che effettua l'integrazione della corrente per determinare una stima del SoC e da un modulo contenente un modello (elettrico) che simula il comportamento della cella; il confronto del comportamento effettivo della cella con l'uscita del modello consente di realizzare una correzione del Coulomb Counting, in modo tale da poter risolvere i problemi che caratterizzano tale metodo (discussi nel capitolo precedente).

La parte relativa all'identificazione dei parametri si occupa di effettuare la stima dei parametri che caratterizzano il modello della batteria in base a misure sulla corrente e tensione di cella. La stima viene effettuata utilizzando il metodo del Moving Window Least Square (MWLS): si seleziona una finestra temporale all'interno della quale vengono acquisiti i dati della cella, con i quali, tramite una stima nel senso dei minimi quadrati (Least Square, LS), vengono determinati i nuovi parametri del modello. Dunque il modello della cella viene aggiornato in tempo reale, e risulta dunque più accurato di un semplice modello a parametri costanti.
La qualità della stima LS dipende in maniera significativa dalla qualità degli ingressi, i quali devono avere un buon rapporto segnale-rumore e devono avere delle componenti di segnale consistenti nell'intervallo di frequenze in cui il sistema andrà ad operare. Per queste ragioni può essere utile filtrare i segnali (i_L e v_T) con un filtro passa-basso, in modo tale da limitare la banda dei segnali sulla banda di interesse e ridurre le componenti di rumore in alta frequenza.

Un altro aspetto molto importante del processo di stima e di identificazione dei parametri è la scelta della frequenza di campionamento. Poiché le dinamiche delle grandezze in questione sono "lente" (le costanti di tempo degli effetti di rilassamento sono infatti dell'ordine di secondi o di decine di secondi), la frequenza di campionamento non deve essere elevata: una frequenza di campionamento effettiva di qualche Hz è sufficiente. Ciò significa che oltre al filtraggio del segnale, sarà necessario un blocco di decimazione che riduca il numero di campioni acquisiti.

La finestra temporale di dati su cui viene effettuata l'individuazione dei parametri deve essere opportunamente controllata in modo tale che i campioni acquisiti in tale intervallo risultino adatti a consentirne una corretta identificazione. Può infatti accadere che in certe circostanze una lunghezza fissa del numero di campioni da acquisire non permetta di ottenere una corretta stima dei parametri; in particolare ciò avviene quando la matrice $\Phi^{\mathrm{T}}\Phi$, di cui parleremo più avanti, non ha rango pieno, ed è quindi necessario acqui-

sire ancora dati (finché non si raggiunge la condizione di rango pieno). Per risolvere questo problema è presente il modulo di *window check*, un controllo che determina se i dati acquisiti consentono di ottenere una identificazione corretta, e stabilisce quindi se i parametri debbano essere aggiornati oppure no, come mostrato in Figura 3.1.

3.2 Modello circuitale

Come già discusso nel capitolo 2, in letteratura si possono trovare un gran numero di modelli di batterie caratterizzati da diversi gradi di complessità. Tra di essi, i modelli elettrici sono più semplici ed intuitivi, e possono essere direttamente simulati con programmi di simulazione elettrica, senza dover effettuare una sintesi matematica del modello. Per queste ragioni è stato scelto di utilizzare un modello circuitale.
Il modello circuitale impiegato è riportato in Figura 3.2.
Questo modello è una variante del modello circuitale presentato nel para-

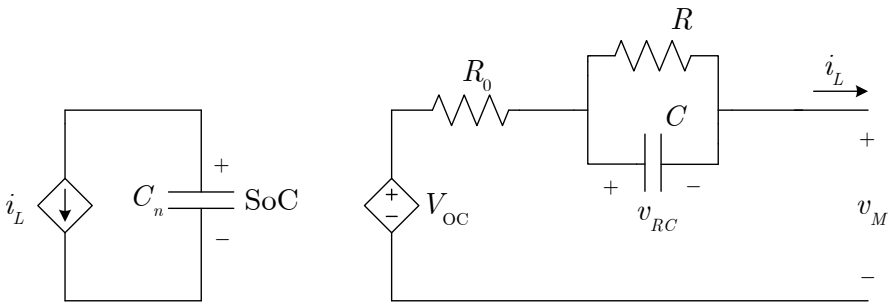

Figura 3.2: Circuito elettrico equivalente (modello ridotto)

grafo 2.3.4 con un unico gruppo RC, ed è indicato come *modello ridotto*; il *modello completo* è invece costituito da due gruppi RC. Due gruppi RC sono infatti sufficienti a simulare accuratamente gli effetti transitori delle batterie al litio [17]. Nonostante il modello completo sia più accurato di quello ridotto, si possono raggiungere comunque buoni risultati anche con l'utilizzo di quest'ultimo, in particolare nel caso di transitori veloci [11, 18].

Le equazioni nel dominio del tempo che caratterizzano il modello sono riportate nel sistema 3.1:

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\mathrm{SoC}(t) = -\dfrac{i_L(t)}{Q_R} \\ \dfrac{\mathrm{d}}{\mathrm{d}t}v_{RC}(t) = -\dfrac{v_{RC}(t)}{RC} + \dfrac{i_L(t)}{C} \\ v_M(t) = V_{\mathrm{OC}} - R_0 i_L(t) - v_{RC}(t) \end{cases} \tag{3.1}$$

La corrente i_L, ovvero la corrente di carico della cella, costituisce l'ingresso del modello, mentre v_M è la tensione di uscita del modello che verrà confrontata con la tensione v_T misurata ai terminali della cella. i_L e v_T sono dunque due grandezze note in quanto vengono direttamente misurate, e costituiscono le uniche due variabili accessibili del sistema. Q_R è la carica massima estraibile dalla cella.

La tensione a circuito aperto (V_{OC}) dipende dal valore dello stato di carica tramite una relazione non lineare $V_{\mathrm{OC}} = f(\mathrm{SoC})$; dunque nel modello è presente una componente non lineare che accresce la complessità di analisi della stima. Tale relazione non lineare può essere espressa per mezzo di equazioni non lineari, come ad esempio quelle proposte in [15]. Alcune di esse considerano anche l'effetto di isteresi della batteria, che nel caso in questione (batterie di tipo NMC), risulta trascurabile. Inoltre, sebbene modelli di questo tipo possano essere accurati, la conversione di una caratteristica sperimentale SoC-OCV in sistemi di equazioni comporta inevitabilmente degli errori.

E' stato dunque scelto in questo progetto di utilizzare una look-up table (LUT) per ricostruire tale relazione; i valori contenuti nella LUT sono stati ricavati da off-line tests, come verrà esposto in dettaglio nel capitolo successivo. Una soluzione di questo tipo presuppone che la relazione SoC-OCV sia costante nel tempo e che non dipenda (o al limite abbia una dipendenza trascurabile) dalle condizioni di utilizzo della batteria; in letteratura sono stati effettuati degli studi in proposito [19, 20] e i risultati confermano la stabilità della caratteristica.

Nonostante la non linearità della curva di SoC-OCV, poiché per valori di

corrente di carica/scarica ordinari il SoC sperimenta "piccole" variazioni, si può effettuare una linearizzazione della caratteristica SoC-OCV [18] intorno al punto di lavoro, per cui si ha:

$$V_{\mathrm{OC}} = f(\mathrm{SoC}) = \alpha_0 + \alpha_1 \cdot \mathrm{SoC} \tag{3.2}$$

3.3 Algoritmo di stima del SoC

L'algoritmo di stima del SoC realizzato prevede l'utilizzo contemporaneo di un modulo di integrazione della corrente (Coulomb Counting) per ottenere una prima stima del SoC affetta da errore e di un modello circuitale della cella tramite il quale è possibile correggere questo errore per effetto di un meccanismo di retro-azione. In Figura 3.3 è riportato lo schema a blocchi del sistema con cui è realizzato l'algoritmo di stima.

Il sistema è costituito da due ingressi, ovvero la corrente (i_L) e la tensione

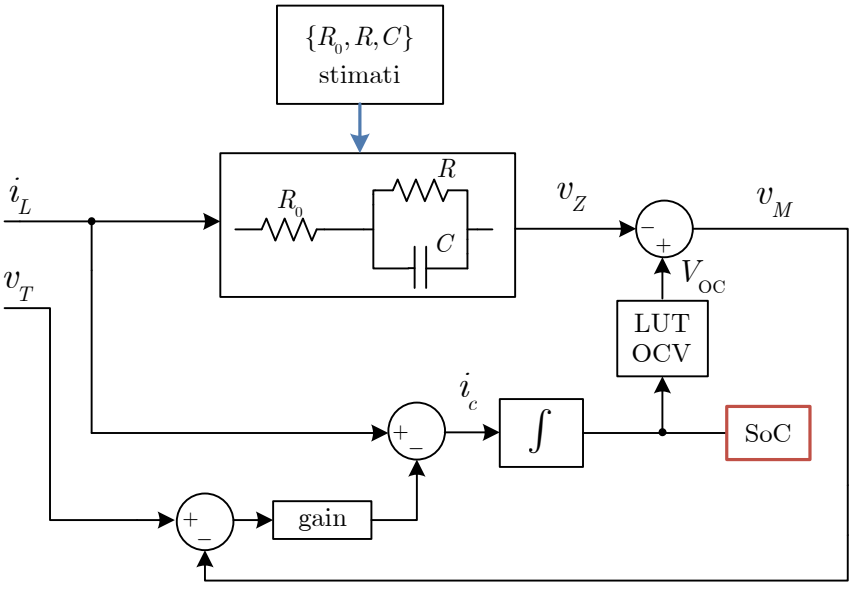

Figura 3.3: Schema a blocchi del sistema di stima del SoC

(v_T) misurate, e da un'uscita, il SoC. La corrente i_L è inviata in ingresso al

modulo di integrazione da cui viene determinata una stima del SoC:

$$\mathrm{SoC}(t) = \mathrm{SoC}_{\mathrm{init}} + \frac{1}{Q_R}\int_{t_0}^{t} i_c(\tau)\,\mathrm{d}\tau \tag{3.3}$$

Il SoC ricavato con questa operazione è tuttavia affetto dagli errori tipici del metodo di Coulomb Counting già descritti in precedenza.
Il SoC così estratto viene inviato in ingresso alla LUT contenente i punti della caratteristica SoC-OCV, la quale fornisce in uscita il valore dell'OCV corrispondente. Questo valore è utilizzato dal modello per il calcolo della tensione v_M, la quale rappresenta una stima della tensione di cella v_T.
Il modello è periodicamente aggiornato con i nuovi valori dei parametri determinati dal processo di identificazione on-line; la sua uscita è la tensione v_M, la quale rappresenta una stima della tensione di cella. Sottraendo dunque v_M al valore reale (misurato) della tensione di cella v_T si ottiene l'errore sulla stima della tensione. Tale errore è moltiplicato per un opportuno coefficiente di guadagno e il risultato è sottratto dalla corrente i_L. Infine, il risultato della sottrazione delle due correnti viene inviato al modulo di Coulomb Counting. L'algoritmo di stima del SoC è dunque realizzato da un sistema in retroazione con un controllo di tipo proporzionale che consente di ottenere una stima del SoC che non presenta gli svantaggi caratteristici dei metodi di Coulomb Counting e della stima con modello circuitale considerati singolarmente, per cui la stima risulta essere molto più accurata.

3.4 Stabilità del sistema e sensibilità agli errori

Il sistema descritto è un sistema in retroazione caratterizzato dalla presenza di un controllore proporzionale il cui guadagno (che indicheremo con K_P) deve ancora essere definito. Esso dovrà essere scelto in modo tale da garantire la stabilità del sistema e da consentire, se possibile, una correzione degli errori che caratterizzano le misure delle grandezze di ingresso v_T e i_L. In seguito viene brevemente esposta l'analisi sulla stabilità del sistema e il suo comportamento in presenza di errori.

3.4.1 Analisi sulla stabilità

L'analisi di stabilità dell'algoritmo si effettua nel dominio della frequenza, ovvero da un punto di vista dinamico dopo aver linearizzato la relazione SoC-OCV. In accordo con il principio di sovrapposizione degli effetti è possibile valutare separatamente il contributo di ciascun ingresso sull'uscita (il SoC): se per ciascun ingresso la funzione di trasferimento risulta stabile, allora il sistema è stabile.

Dallo schema a blocchi di Figura 3.4, dove si considera la forma linearizzata

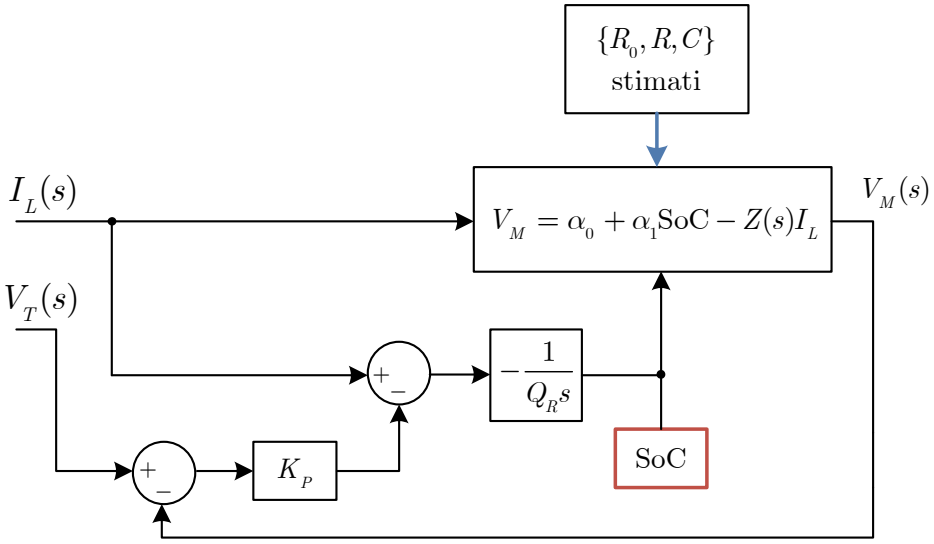

Figura 3.4: Schema a blocchi nel dominio di Laplace

della caratteristica SoC-OCV descritta dalla (3.2), si ricavano le funzioni di trasferimento del sistema applicando la sovrapposizione degli effetti. Da notare che in un'analisi alle variazioni il contributo di α_0 non viene preso in considerazione. Dunque si ha:

$$\begin{aligned} \mathrm{SoC}(s)|_{i_L=0} &= -\frac{1}{Q_R s}\left[-V_T(s)K_P + \alpha_1 \mathrm{SoC}(s) K_P\right] = \\ &= \frac{K_P}{Q_R s}\left[V_T(s) - \alpha_1 \mathrm{SoC}(s)\right] \end{aligned}$$

da cui

$$\frac{\text{SoC(s)}}{V_T(s)} = \frac{\dfrac{K_P}{Q_R}}{s + \dfrac{\alpha_1 K_P}{Q_R}} \equiv W_V(s) \tag{3.4}$$

Per ottenere l'altra funzione di trasferimento (con ingresso i_L) si pone $v_T = 0$:

$$\begin{aligned}\text{SoC}(s)|_{v_T=0} &= -\frac{1}{Q_R s}\left[I_L(s) + \alpha_1 K_P \text{SoC}(s) - K_P Z(s) I_L(s)\right] = \\ &= -\frac{I_L}{Q_R s}[1 - K_P Z(s)] - \frac{\alpha_1 K_P}{Q_R s}\text{SoC}(s)\end{aligned}$$

dove

$$Z(s) = R_0 + \frac{R}{1 + RCs} = \frac{(R_0 + R) + R_0 RCs}{1 + RCs}$$

la funzione di trasferimento con ingresso i_L dunque risulta:

$$\frac{\text{SoC(s)}}{I_L(s)} = \frac{\dfrac{1}{Q_R}[K_P Z(s) - 1]}{s + \dfrac{\alpha_1 K_P}{Q_R}} \equiv W_I(s) \tag{3.5}$$

Poiché le due funzioni di trasferimento hanno lo stesso denominatore, e quindi gli stessi poli, per entrambi gli ingressi la stabilità del sistema dipende dal segno del termine $\alpha_1 K_P / Q_R$. Dato che α_1, che rappresenta la derivata della caratteristica SoC-OCV nel punto di lavoro, è sempre positivo e Q_R, che rappresenta la carica massima estraibile dalla cella, è anch'esso positivo, risulta evidente che la stabilità dipenda esclusivamente dal segno di K_P. Si hanno i seguenti casi:

- $K_P = 0$: in questo caso il sistema di retroazione viene annullato ed il sistema opera ad anello aperto, dunque il SoC stimato coincide con il SoC calcolato tramite il metodo del Coulomb Counting.
- $K_P > 0$: per valori positivi di K_P il sistema risulta asintoticamente stabile.

Dunque per la stabilità del sistema è necessario scegliere $K_P > 0$.

3.4.2 Risposta in presenza di errori di misura

Le misure dei due ingressi sono inevitabilmente affette da errori, che, com'è noto, possono essere suddivisi in una componente costante, l'offset, ed in una componente variabile, il rumore. In questa trattazione viene analizzata la risposta del sistema in presenza di un errore di misura costante, ovvero dovuto alla sola presenza di un offset.

Consideriamo dunque il segnale di ingresso v_T a cui si aggiunge una componente di errore costante V_{err}. Si applica anche in questo caso il principio di sovrapposizione degli effetti: si considera la risposta del sistema all'ingresso V_{err}, ovvero la risposta ad un ingresso a gradino di ampiezza pari a V_{err}:

$$\text{SoC}_{\text{err}}(s) = \frac{V_{\text{err}}}{s} W_V(s) = \frac{V_{\text{err}}}{s} \frac{\dfrac{K_P}{Q_R}}{s + \dfrac{\alpha_1 K_P}{Q_R}}$$

Per valutare il comportamento del sistema in presenza di tale errore si determina il valore di SoC_{err} a regime utilizzando il teorema del valore finale:

$$\lim_{t\to\infty} \text{SoC}_{\text{err}}(t) = \lim_{s\to 0} s\,\text{SoC}_{\text{err}}(s) = \lim_{s\to 0} s \frac{V_{\text{err}}}{s} \frac{\dfrac{K_P}{Q_R}}{s + \dfrac{\alpha_1 K_P}{Q_R}} = \frac{V_{\text{err}}}{\alpha_1}$$

Si nota dunque che il sistema non è in grado di correggere queste tipologie di errori; inoltre, il risultato dell'operazione di limite non dipende da K_P ma solo da α_1 che è un parametro caratteristico della cella su cui non si può agire. Tale valore di fatto non è costante ma dipende dallo stato di carica della cella dato che la relazione SoC-OCV non è lineare; ciò significa che nella zona in cui la curva è "più piatta", ovvero ha pendenza minore, il contributo di errore sulla stima del SoC dovuto ad un errore sulla misura della tensione sarà più significativo.

Per quanto riguarda invece il comportamento del sistema in presenza di

un errore sulla corrente, I_{err}, si ha:

$$\text{SoC}_{\text{err}}(s) = \frac{I_{\text{err}}}{s} W_I(s) = \frac{I_{\text{err}}}{s} \frac{\frac{K_P}{Q_R}[Z(s) - 1]}{s + \frac{\alpha_1 K_P}{Q_R}}$$

applicando il teorema del valore finale si ottiene:

$$\lim_{t\to\infty} \text{SoC}_{\text{err}}(t) = \lim_{s\to 0} s\, \text{SoC}_{\text{err}}(s) = \lim_{s\to 0} s \frac{I_{\text{err}}}{s} \frac{\frac{1}{Q_R}[K_P Z(s) - 1]}{s + \frac{\alpha_1 K_P}{Q_R}} =$$

$$= I_{\text{err}} \frac{K_P Z(0) - 1}{\alpha_1 K_P}$$

dove $Z(0) = R_0 + R_1$.
In questo caso si ha che l'effetto di I_{err} sulla stima del SoC dipende dal valore di K_P. In particolare, si nota che se $K_P = 0$ (algoritmo equivalente al solo Coulomb Counting) si ha che l'errore sul SoC tende a $-\infty$ se $I_{\text{err}} > 0$ a $+\infty$ se $I_{err} < 0$, com'era logico aspettarsi in quanto in ingresso all'integratore è presente una componente di errore costante che tende a farne divergere l'uscita; se invece $K_P \to +\infty$ (algoritmo equivalente al solo modello circuitale) l'errore tende ad una quantità costante e positiva pari a $I_{err}\, Z(0)/\alpha_1$. Dunque esiste un valore di K_P che annulla l'errore sulla stima del SoC dovuto ad un offset sulla corrente:

$$K_P Z(0) - 1 = 0 \qquad \to \qquad K_{P_{\text{opt}}} = \frac{1}{Z(0)} = \frac{1}{R_0 + R}$$

Da notare che il valore di $K_{P_{\text{opt}}}$ non è costante in quanto i valori di R_0 e R vengono aggiornati periodicamente tramite l'identificazione on-line dei parametri.

3.4.3 Risposta ad un errore sul valore iniziale del SOC

Come già accennato in precedenza, il metodo del Coulomb Counting necessita una conoscenza del valore iniziale del SoC, SoC_{init}, che sia la più accurata possibile, in quanto un errore sul suo valore non potrebbe essere recuperato.

Inoltre non è semplice calcolare il valore del SoC in maniera accurata a meno di non essere in uno stato noto (tipicamente questo stato è lo stato di cella completamente carica).
Si analizza dunque il comportamento del sistema proposto in presenza di un errore sul valore iniziale del SoC. Considerando lo schema a blocchi di Figura 3.4, si applica un segnale di errore, $\mathrm{SoC}_{\mathrm{init}_{\mathrm{err}}}$ in ingresso al modulo di integrazione; l'uscita relativa a tale ingresso è

$$\mathrm{SoC}(s) = \mathrm{SoC}_{\mathrm{init}_{\mathrm{err}}} - \frac{1}{Q_R s}\,\alpha_1 K_P \mathrm{SoC}(s)$$

da cui

$$\frac{\mathrm{SoC}(s)}{\mathrm{SoC}_{\mathrm{init}_{\mathrm{err}}}} = \frac{s}{s + \dfrac{K_P}{Q_R}\alpha_1}$$

applicando il teorema del valore finale si ha:

$$\lim_{t\to\infty} \mathrm{SoC}_{\mathrm{err}}(t) = \lim_{s\to 0} s\,\mathrm{SoC}_{\mathrm{err}}(s) = \lim_{s\to 0} s\,\frac{\mathrm{SoC}_{\mathrm{init}_{\mathrm{err}}}}{s}\,\frac{s}{s + \dfrac{K_P}{Q_R}\alpha_1} = 0$$

Dunque il sistema proposto è in grado di compensare l'errore sul valore iniziale del SoC; tale correzione avviene tramite un andamento esponenziale con costante di tempo $K_P/Q_R\,\alpha_1$.

3.5 Stima on-line dei parametri

Il modulo di identificazione on-line dei parametri ricava una stima dei parametri a partire da un certo numero di campioni delle grandezze i_L e v_T presi in una finestra temporale limitata, utilizzando il metodo del Moving Window Least Square (MWLS) [21].
Per identificare i parametri di un sistema lineare tempo-invariante (LTI) occorre descrivere il sistema attraverso un opportuno modello, che, nel caso in questione, è costituito dalla struttura ARX (*Autoregressive Exogenous Model*), come descritto in [11, 18, 21, 22]:

$$A(q)\,y(q) = B(q)\,u(q) + e(q) \tag{3.6}$$

dove

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + \cdots + a_n q^{-n} \\ B(q) &= b_0 + b_1 q^{-1} + \cdots + b_m q^{-m} \end{aligned} \tag{3.7}$$

e $e(q)$ è il rumore gaussiano bianco.
Con questo modello le uscite ad un certo istante possono essere stimate a partire dal valore dei campioni degli ingressi agli istanti precedenti. Tramite il metodo LS (Least-Square) è possibile minimizzare l'errore quadratico medio tra il valore dell'uscita stimato e il valore dell'uscita all'istante presente. Nel caso in cui i campioni di ingresso/uscita siano aggiornati durante l'esecuzione, è inoltre possibile utilizzare il metodo Moving Window Least Square. In questo metodo la stima è effettuata applicando il metodo LS su L dati acquisiti durante l'intervallo temporale $[t - L,\, t]$.

I parametri della batteria da identificare sono: $\{\alpha_0,\, \alpha_1,\, R_0,\, R,\, C\}$. Avendo in precedenza linearizzato il modello della cella, possiamo adesso ottenere la funzione di trasferimento considerando la corrente i_L come grandezza di ingresso e la tensione v_M come grandezza di uscita, utilizzando la trasformata di Laplace sul sistema di equazioni (3.1) riportato di seguito per semplicità:

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\mathrm{SoC}(t) = -\dfrac{i_L(t)}{Q_R} \\ \dfrac{\mathrm{d}}{\mathrm{d}t} v_{RC}(t) = -\dfrac{v_{RC}(t)}{RC} + \dfrac{i_L(t)}{C} \\ v_M(t) = \alpha_0 + \alpha_1 \cdot \mathrm{SoC}(t) - R_0 i_L(t) - v_{RC}(t) \end{cases}$$

applicando la trasformata di Laplace:

$$\begin{cases} s\,\mathrm{SoC}(s) = -\dfrac{I_L(s)}{Q_R} \\ s\,V_{\mathrm{RC}}(s) = -\dfrac{V_{RC}(s)}{RC} + \dfrac{I_L(s)}{C} \\ V_M(s) = \alpha_0 + \alpha_1 \cdot \mathrm{SoC}(s) - R_0 I_L(s) - V_{RC}(s) \end{cases} \tag{3.8}$$

da cui si ottiene:

$$\begin{aligned} \mathrm{SoC}(s) &= -\frac{I_L(s)}{s\,Q_R} \\ V_{\mathrm{RC}}(s) &= \frac{I_L(s)}{1+RCs} \end{aligned} \tag{3.9}$$

Sostituendo le (3.9) nella terza equazione del sistema (3.8) si ha:

$$V_M(s) = \alpha_0 - \alpha_1 \frac{I_L(s)}{s\,Q_R} - R_0 I_L(s) - \frac{I_L(s)}{1+RCs} \tag{3.10}$$

e infine indicando con $(Y(s),\, U(s))$ le variabili di ingresso/uscita del sistema rispettivamente corrispondenti alla trasformata di Laplace di v_M e i_L, si ottiene:

$$\frac{Y(s)-\alpha_0}{U(s)} = \frac{R_0 s^2 + \left(\frac{\alpha_1}{Q_R} + \frac{1}{C} + \frac{R_0}{RC}\right) s + \frac{\alpha_1}{Q_R RC}}{\left(s + \frac{1}{RC}\right) s} \tag{3.11}$$

Per riportarsi nel dominio discreto (ovvero nel dominio della variabile z) si utilizza la trasformazione di Tustin, che consente consiste in una integrazione trapezoidale, effettuando la sostituzione:

$$s \to \frac{2}{T}\,\frac{1-z^{-1}}{1+z^{-1}}$$

dopo alcuni passaggi [16] si ottiene:

$$\frac{Y(z^{-1})-\alpha_0}{U(z^{-1})} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{3.12}$$

dove:

$$
\begin{aligned}
b_0 &= \frac{2\alpha_0 TRC - \alpha_1 T^2 + 2Q_R RT - 4Q_R R_0 RC}{2Q_R T + 4Q_R RC} \\
b_1 &= \frac{-\alpha_1 T^2 + 4Q_R R_0 RC}{Q_R T + 2Q_R RC} \\
b_2 &= \frac{-2\alpha_1 TRC - b_1 T^2 - 2Q_R RT - 4Q_R R_0 RC - 2Q_R R_0 T}{2Q_R T + 4Q_R RC} \\
a_1 &= \frac{-4RC}{2RC + T} \\
a_2 &= \frac{2RC - T}{2RC + T}
\end{aligned}
\tag{3.13}
$$

In accordo con l'espressione (3.7) la relazione nel dominio del tempo tra i campioni di ingresso/uscita può essere scritta nella seguente forma:

$$
\begin{aligned}
y(k) = &- a_1 y(k-1) - a_2 y(k-2) + \alpha_0(1 + a_1 + a_2) \\
&+ b_0 u(k) + b_1 u(k-1) + b_2 u(k-2)
\end{aligned}
\tag{3.14}
$$

dove

$$
1 + a_1 + a_2 = 0 \tag{3.15}
$$

Tale relazione è infatti imposta dalle equazioni 4 e 5 delle (3.13). Questo significa che il valore di α_0 non ha alcun effetto sulla stima di $y(k)$ e, di conseguenza, degli altri parametri. In altri termini, α_0 ha la funzione di un offset sull'uscita che non influenza il comportamento del sistema.
L'espressione (3.14) dei campioni di ingresso/uscita sarà impiegata in seguito per effettuare l'identificazione dei parametri utilizzando il metodo del MWLS esposto più avanti.
Supponendo di aver già effettuato la stima dei parametri e di aver quindi determinato $\{a_1,\, a_2,\, b_0,\, b_1,\, b_2\}$, per risalire ai valori di $\{R_0,\, R,\, C\}$ si procede come segue. Risolvendo le equazioni (3.13) si ottiene un' espressione univoca dei parametri della batteria in funzione dei coefficienti $\{a_1,\, a_2,\, b_0,\, b_1,\, b_2\}$. In

particolare dalla quarta equazione delle (3.13) si ottiene:

$$R = -\frac{T}{2C}\frac{a_1}{2+a_1} \tag{3.16}$$

Sostituendo poi il valore di R nelle prime tre equazioni delle (3.13) si ricava il seguente sistema:

$$\begin{cases} X1 = T^2\left(-1+\dfrac{a_1}{2+a_1}\right)\alpha_1 + 2Q_RT\left(-1+\dfrac{a_1}{2+a_1}\right)R_0 + Q_RT^2\left(\dfrac{a_1}{2+a_1}\right)\dfrac{1}{C} \\ X2 = -2T^2\alpha_1 - 4Q_RT\left(\dfrac{a_1}{2+a_1}\right)R_0 \\ X3 = -T^2\left(1+\dfrac{a_1}{2+a_1}\right)\alpha_1 + 2Q_RT\left(1+\dfrac{a_1}{2+a_1}\right)R_0 - Q_RT^2\left(\dfrac{a_1}{2+a_1}\right)\dfrac{1}{C} \end{cases} \tag{3.17}$$

dove

$$X1 = 2Q_RT\left(1-\frac{a_1}{2+a_1}\right)b_0$$

$$X2 = 2Q_RT\left(1-\frac{a_1}{2+a_1}\right)b_1$$

$$X3 = 2Q_RT\left(1-\frac{a_1}{2+a_1}\right)b_2$$

Utilizzando le (3.17), assieme alle relazioni (3.15) e (3.16), si può determinare il set di parametri $\{\alpha_0, \alpha_1, R_0, R, C\}$.

Come già accennato in precedenza, l'identificazione dei parametri viene effettuata con il metodo MWLS. Esso consiste nel metodo LS applicato ad un insieme di dati che non sono statici ma vengono continuamente aggiornati. La stima dei parametri è dunque effettuata per ciascun istante di tempo sfruttando un insieme di valori degli ingressi e delle uscite precedentemente acquisiti in una finestra temporale (window). Questo metodo può dunque essere schematizzato come segue:

1. all'istante k al modulo di identificazione dei parametri vengono inviati

i dati di ingresso (ovvero i campioni di tensione e corrente) acquisiti durante l'intervallo temporale $[k - L + 1,\, k - L + 2, \ldots, k]$, scegliendo opportunamente il valore della lunghezza L della finestra, ovvero in modo tale da consentire una identificazione corretta dei parametri;

2. si effettua la stima dei parametri con il metodo LS applicato ai campioni acquisiti;
3. si sposta la finestra temporale di un certo valore temporale, N. Si aggiorna quindi la finestra temporale e si riparte dal punto 1.

Vediamo adesso come avviene l'identificazione dei parametri con il metodo MWLS. Come visto in precedenza, si considera un sistema descritto dall'equazione (3.14), dove u e y rappresentano rispettivamente l'ingresso i_L e l'uscita v_T del sistema. Per determinare l'identificazione dei parametri ad un certo istante t si sceglie la lunghezza L della finestra temporale di identificazione e si raccolgono in due vettori $\underline{y}$ e $\underline{u}$ i campioni di y e u acquisiti nell'intervallo temporale compreso tra $t - L + 1$ e t.

$$\underline{y} = [y(t - L + 1), \ldots, y(t)]^{\mathrm{T}}$$
$$\underline{u} = [u(t - L + 1), \ldots, u(t)]^{\mathrm{T}}$$

Poiché la relazione (3.14) può anche essere espressa come

$$y(t) = \varphi(t)\underline{\theta}$$

dove

$$\varphi(t) = [-y(t-1) \quad -y(t-2) \;\; u(t) \;\; u(t-1) \;\; u(t-2)]$$
$$\underline{\theta} = [a_1 \;\; a_2 \;\; b_0 \;\; b_1 \;\; b_2]^{\mathrm{T}}$$

con gli L campioni acquisiti si può costruire una matrice Φ di dimensione

$L \times 5$ le cui righe sono costituite da $\varphi(t)$:

$$\begin{aligned}
\varphi(t-L+1) &= [0 \;\; 0 \;\; u(t-L+1) \;\; 0 \;\; 0] \\
\varphi(t-L+2) &= [y(t-L+1) \;\; 0 \;\; u(t-L+2) \;\; u(t-L+1) \;\; 0] \\
&\vdots \\
\varphi(t) &= [-y(t-1) \;\; -y(t-2) \;\; u(t) \;\; u(t-1) \;\; u(t-2)]
\end{aligned}$$

per cui il sistema può essere descritto come:

$$\underline{y} = \Phi\,\underline{\theta} \tag{3.18}$$

dove

$$\Phi = \begin{bmatrix} \varphi(t-L+1) \\ \varphi(t-L+2) \\ \vdots \\ \varphi(t) \end{bmatrix}$$

Adesso che il sistema è stato espresso nella forma (3.18) si può applicare una delle varie tecniche di risoluzione dei problemi LS per determinare un'approssimazione di $\underline{\theta}$ nel senso dei minimi quadrati ed ottenere così una stima dei parametri della cella. Nel seguito del capitolo saranno descritte le tecniche di risoluzione dei problemi LS con particolare riferimento alla loro implementazione su FPGA.

3.6 Algoritmi di risoluzione di problemi dei minimi quadrati

In questo paragrafo sono presentati alcuni dei metodi di risoluzione dei problemi di LS più utilizzati, la cui conoscenza è indispensabile per effettuare una scelta sul metodo di risoluzione del problema di MWLS per l'identificazione dei parametri della cella. Per ciascuno dei metodi presentati viene svolta un stima sulla complessità in termini di *flops* (floating point operations) e viene effettuato un confronto tra di essi, con particolare riferimento alla loro implementabilità su FPGA.

Gli FPGA sono infatti sempre più utilizzati per applicazioni di tipo digital signal processing: sebbene molto spesso essi non riescano a raggiungere le frequenze di clock di processori general-purpose o di DSPs (Digital Signal Processors), in applicazioni in cui è possibile realizzare una significativa parallelizzazione del flusso di dati gli FPGA consentono di ottenere prestazioni migliori. Il problema è che spesso essi non forniscono buoni risultati in termini di rapporto prestazioni/costi per calcoli in virgola mobile (singola o doppia precisione) a causa di latenze nel processing e congestione del routing, oltre al fatto che il flusso di progetto tradizionale di un FPGA prevede di effettuare una descrizione hardware RTL (Register-Transfer-Level) in Verilog o VHDL che non si adatta bene al trattamento di dati in virgola mobile [23, 24].
Nel caso di una applicazione come quella in questione può essere realizzata una parallelizzazione del flusso di dati in quanto si può pensare di progettare un modulo di stima del SoC che esegua la stima di tutte le celle della batteria. Il modulo di stima del SoC può infatti lavorare indipendentemente dal resto e quindi in modo parallelo, svincolandosi dal software, il quale può quindi occuparsi di compiere le altre funzionalità del BMS (una possibile architettura di un BMS realizzabile su FPGA sarà presentata nel successivo capitolo). Inoltre, utilizzando FPGA Altera e il tool di sviluppo DSP builder, descritto nel capitolo successivo, è possibile velocizzare e semplificare il flusso di progetto, rendendolo più adatto ad operare con dati in virgola mobile. Ciò giustifica le scelte progettuali effettuate e in particolare la necessità di determinare un buon algoritmo per la risoluzione dei problemi di LS che sia implementabile in FPGA.

Sono di seguito presentati due metodi di risoluzione di sistemi di equazioni nel senso dei minimi quadrati, ciascuno dei quali presenta due algoritmi di fattorizzazione. Tali algoritmi sono stati scelti per la loro adattabilità ad un'implementazione su FPGA. Essi (ed altri algoritmi) sono descritti in dettaglio in [25].

3.6.1 Problema dei minimi quadrati

Si consideri il problema di determinare un vettore $x \in \mathbb{R}^n$ tale che $Ax = b$, dove $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$ sono noti e $m \geq n$. Poiché vi sono più equazioni che incognite, il sistema è detto *sovra-determinato*. Solitamente un sistema di questo tipo non ha soluzione, per cui si cerca di minimizzare la quantità $\| Ax - b \|_2$. Il problema dei minimi quadrati consiste dunque nel determinare

$$\min_{x \in \mathbb{R}^n} \| Ax - b \|_2$$

Tale problema può essere risolto con due diversi approcci. Il primo è la *fattorizzazione QR*, che consiste nell'effettuare una scomposizione della matrice A nel prodotto di due matrici: una matrice ortonormale $Q \in \mathbb{R}^{m \times m}$ e una matrice triangolare superiore $R \in \mathbb{R}^{m \times n}$ (ovvero tale che la sua sottomatrice $n \times n$ più in alto è una matrice triangolare superiore) tali che:

$$A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

Dunque, supponendo di aver determinato una fattorizzazione QR della matrice A si ha che:

$$A = QR \rightarrow Q^{\mathrm{T}} A = R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \begin{matrix} n \\ m - n \end{matrix}$$

e posto

$$Q^{\mathrm{T}} b = \begin{bmatrix} c \\ d \end{bmatrix} \begin{matrix} n \\ m - n \end{matrix}$$

allora la soluzione nel senso dei minimi quadrati si determina risolvendo

$$R_1 x = c$$

E' da notare che A può essere scomposta anche come prodotto tra una matrice $m \times n$ ortonormale e una matrice $n \times n$ triangolare superiore; tale

fattorizzazione è detta *thin QR factorization*. Si osserva infatti che:

$$A = QR = \left[\begin{array}{c|c} Q_1 & Q_2 \end{array} \right] \left[\begin{array}{c} R_1 \\ 0 \end{array} \right] = Q_1 R_1$$

dove $Q_1 \in \mathbb{R}^{m \times n}$ e $Q_2 \in \mathbb{R}^{m \times (m-n)}$.

Il secondo metodo consiste invece nel determinare la fattorizzazione di una matrice quadrata $n \times n$; l'algoritmo Cholesky e le sue varianti sono tra gli algoritmi più efficienti per determinare tale fattorizzazione. Supponendo di avere un sistema $Bx = y$ dove $B \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ e con $m \geq n$, esso risulta essere sovra-determinato. Il problema della determinazione della soluzione nel senso dei minimi quadrati può essere trasformato nel seguente problema:

$$\underbrace{B^{\mathrm{T}}B}_{A} x = \underbrace{B^{\mathrm{T}}y}_{b} \rightarrow Ax = b \tag{3.19}$$

dove $A \in \mathbb{R}^{n \times n}$ simmetrica e semi-definita positiva (ovvero $(Av) \cdot v \geq 0 \; \forall v \in \mathbb{R}^n$) e $b \in \mathbb{R}^n$. La soluzione x del sistema $Ax = b$ è la soluzione nel senso dei minimi quadrati del sistema. Fattorizzando A in maniera opportuna il calcolo di x risulta immediato e non è necessario ricavare l'inversa della matrice A.

Di seguito sono riportati alcuni algoritmi di fattorizzazione ed è mostrato come applicare tali fattorizzazioni si applicano alla determinazione della soluzione del problema LS. Per ciascun algoritmo viene riportato il codice in un linguaggio "matlab-like" e viene presentato un calcolo del numero di flops ([25], p.12) per valutarne la complessità; inoltre vengono fatte considerazioni sulla possibile implementazione di ciascun algoritmo su FPGA.

3.6.2 Fattorizzazione QR con l'algoritmo di Gram-Schmidt

L'algoritmo di Gram-Schmidt è uno degli algoritmi più utilizzati per il calcolo della fattorizzazione QR. Esso fornisce una fattorizzazione di $A \in \mathbb{R}^{m \times n}$ in una matrice $Q \in \mathbb{R}^{m \times n}$ e una matrice $R \in \mathbb{R}^{n \times n}$. Il calcolo degli elementi di

Q è il seguente:

$$\begin{aligned} u_1 &= a_1 \qquad\qquad , \quad q_1 = \frac{u_1}{\| u_1 \|_2} \\ u_2 &= a_2 - \frac{q_1 \cdot a_2}{(\| q_1 \|_2)^2} q_1 \quad , \quad q_2 = \frac{u_2}{\| u_2 \|_2} \\ &\vdots \qquad\qquad\qquad\quad \vdots \\ u_k &= a_k - \sum_{j=1}^{k-1} \frac{q_j \cdot a_k}{(\| q_j \|_2)^2} q_j \quad , \quad q_k = \frac{u_k}{\| u_k \|_2} \end{aligned}$$

dove a_i, u_i, q_i sono vettori $\in \mathbb{R}^n$.
Una volta ottenuta la matrice Q, si può procedere al calcolo di R come segue:

$$Q^{\mathrm{T}} A = Q^{\mathrm{T}} Q R = R \quad \rightarrow \quad R = Q^{\mathrm{T}} A$$

Dunque gli elementi di Q e R possono essere scritti come:

$$q_k = a_k - \frac{\sum_{j=1}^{k-1} r_{jk} q_j}{r_{kk}} \tag{3.20}$$

$$r_{jk} = q_j \cdot a_k \qquad \text{con } i = 1 : k-1 \tag{3.21}$$

L'algoritmo di Gram-Schmidt è dunque descritto dal seguente codice:

```
for k = 1 : n
    R(k,k) = ‖A(1 : m, k)‖_2
    Q(1 : m, k) = A(1 : m, k)/R(k,k)
    for j = k + 1 : n
        R(k, j) = Q(1 : m, k)^T A(1 : m, j)
        A(1 : m, j) = A(1 : m, j) − Q(1 : m, k)R(k, j)
    end
end
```

Questo algoritmo sovrascrive la matrice A con Q, mentre la matrice R è memorizzata separatamente.

Una volta ottenuta la fattorizzazione QR di A si effettua una *sostituzione diretta*: l'equazione $Ax = QRx = b$ può essere riscritta come:

$$Rx = Q^{\mathrm{T}}b \qquad \rightarrow \qquad \begin{cases} d = Q^{\mathrm{T}}b \\ Rx = d \end{cases}$$

Si risolve dunque il sistema $d = Q^{\mathrm{T}}b$ nell'incognita d:

for $k = 1 : n$
    $d(k, 1) = Q(1 : m, k)^{\mathrm{T}} b(:, 1)$
end

Infine si effettua la *sostituzione inversa*, risolvendo in x il sistema $Rx = d$:

for $k = n - 1 : 1$
    $x(k, 1) = d(k, 1) - R(k, k + 1 : n)\; x(k + 1 : n, 1)$
end

In questo modo si è ottenuta la soluzione x del sistema $Ax = b$ nel senso dei minimi quadrati utilizzando l'algoritmo di fattorizzazione di Gram-Schmidt. Tale algoritmo, come mostrato in seguito, è più complesso degli altri in quanto richiede molti flops e in quanto necessita l'utilizzo di radici quadrate (nel calcolo della norma).
Determinazione del numero di flops: il calcolo di $R(k, k)$ richiede $2m$ flops + una operazione di radice quadrata, mentre il calcolo di $Q(1 : m, k)$ impiega semplicemente m flops per ciascun ciclo del **for** più esterno. Il calcolo di $R(k, j)$ richiede $2m$ flops per ogni ciclo del **for** più interno, così come il calcolo di $A(1 : m, j)$. In totale il numero di flops dell'algoritmo sarà dunque

dato da:

$$\begin{aligned}\text{flops} &= n(2m+m) + \left(2m\sum_{j=1}^{n-1} j + 2m\sum_{j=1}^{n-1} j\right) = \\ &= 3mn + 4m\left[\frac{n}{2}(n-1)\right] = \\ &\approx 3mn + 2mn^2 \approx 2mn^2\end{aligned}$$

avendo trascurato i termini di ordine inferiore in quanto nel calcolo dei flops si considerano m e n "grandi". A questo calcolo va aggiunto anche il calcolo di n radici quadrate. Un algoritmo di questo tipo risulta, come vedremo, più complesso degli altri sia in termini di flops sia a causa della presenza della radice quadrata, che richiede, a livello di implementazione su FPGA, la presenza di un hardware specifico.
Esempi di architetture basate su questo algoritmo sono [26, 27].

3.6.3 Fattorizzazione QR con Givens rotation

E' un metodo alternativo che consente di calcolare la fattorizzazione QR di una matrice A tramite un metodo iterativo che prevede l'utilizzo di una matrice di rotazione del tipo:

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

dove

$$g_{kk} = 1 \text{ per } k \neq i, j$$

$$g_{ii} = g_{jj} = c = cos(\theta)$$

$$g_{ij} = -g_{ji} = s = sin(\theta)$$

$$0 \text{ altrove}$$

Una matrice di questo tipo è chiaramente ortonormale. Se si moltiplica $G^{\mathrm{T}}(i, j, \theta)$ per una matrice A, solo le righe i e j di A risultano coinvolte. Se invece più semplicemente si moltiplica $G^{\mathrm{T}}(i, j, \theta)$ per un vettore $x \in \mathbb{R}^m$, ovvero $y = G^{\mathrm{T}}(i, j, \theta)\, x$, avremo:

$$y_k = \begin{cases} cx_i - sx_j & k = i \\ sx_i + cx_j & k = j \\ x_k & k \neq i, j \end{cases}$$

da questa relazione si nota che è possibile porre $y_j = 0$ imponendo che:

$$c = \cos(\theta) = \frac{x_i}{\sqrt{x_i^2 + x_j^2}} \qquad , \qquad s = \sin(\theta) = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}} \tag{3.22}$$

Dunque moltiplicando $G^{\mathrm{T}}(i, j, \theta)$ per una matrice A sotto la condizione (3.22) si ottiene una matrice in cui l'elemento (j, i)-esimo è nullo. Applicando G^{T} alla matrice A più volte in modo opportuno, ovvero in modo tale da annullare tutti gli elementi di A al di sotto della diagonale principale, è possibile ottenere una matrice triangolare (che coinciderà con la matrice R). Alla fine dell'algoritmo si otterrà una matrice $R \in \mathbb{R}^{m \times n}$ "triangolare superiore" e una

matrice $Q \in \mathbb{R}^{m \times m}$ ortonormale, ottenuta dal prodotto dei G:

$$R = G^{\mathrm{T}}(m,1)\; G^{\mathrm{T}}(m-1,1)\; \cdots\; G^{\mathrm{T}}(m,n)\; A = \\ = \prod_{j=1}^{n} \prod_{i=1}^{m-j} G^{\mathrm{T}}(m-i+1,j)\; A$$

$$Q = G(m,1)\; G(m-1,1)\; \cdots\; G(m,n) = \\ = \prod_{j=1}^{n} \prod_{i=1}^{m-j} G(m-i+1,j)$$

Per presentare l'algoritmo di Givens è necessario prima presentare la seguente funzione, che calcola i valori di c e s tali che

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

function $[c, s] =$ `givens`(a, b)
    if $b = 0$
        $c = 1; s = 0$
    else
        if $|b| > |a|$
            $\tau = -a/b;\;\; s = 1/\sqrt{1+\tau^2};\;\; c = s\tau$
        else
            $\tau = -b/a;\;\; c = 1/\sqrt{1+\tau^2};\;\; s = c\tau$
        end
    end

Questa funzione richiede 5 flops e una radice quadrata. Si noti che non sono coinvolte le inverse di funzioni trigonometriche.
L'algoritmo di Givens è descritto dal seguente codice:

$Q = I_m$
for $j = 1 : n$
  for $i = m : -1 : (j+1)$
    $[c, s] =$ `givens`$(A(i-1, j), A(i, j))$
    $$A(i-1 : i, j : n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^{\mathrm{T}} A(i-1 : i, j : n)$$
    $$Q(i-1 : i, j : n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} Q(i-1 : i, j : n)$$
  end
end

I passi successivi, necessari per ottenere la soluzione del problema LS, sono del tutto analoghi a quelli visti per la fattorizzazione QR con l'algoritmo di Gram-Schmidt, l'unica differenza sono le dimensioni delle matrici (adesso Q è una matrice quadrata mentre R è una matrice rettangolare).
I flops di questo algoritmo possono essere determinati come segue: la funzione `givens` richiede 5 flops ed una radice quadrata per ogni iterazione, mentre il calcolo delle matrici A e Q, poiché coinvolge soltanto le righe i e j della matrice, richiede un numero di flops pari a quelli di un prodotto matriciale tra una matrice 2×2 e una matrice $2 \times (n-j)$, ovvero $2(n-j)$ flops.
I flops totali pertanto risultano:

$$\begin{aligned}
\text{flops} &= \sum_{j=1}^{n} \sum_{i=j+1}^{m} 5 + \sum_{j=1}^{n} \sum_{i=j+1}^{m} 2(n-j) + \sum_{j=1}^{n} \sum_{i=j+1}^{m} 2(n-j) = \\
&= 5 \sum_{j=1}^{n} (m-j) + 4 \sum_{j=1}^{n} [(n-j)(m-j)] = \\
&= 5 \sum_{j=1}^{n} m - 5 \sum_{j=1}^{n} j + 2 \sum_{j=1}^{n} (nm - jm - jn + j^2) = \\
&= 5mn - 5n/2(n-1) + 4 \left(\sum_{j=1}^{n} nm - m \sum_{j=1}^{n} j - n \sum_{j=1}^{n} j + \sum_{j=1}^{n} j^2 \right) \approx \\
&\approx 4n^2 m - 4mn^2/2 - 4nn^2/2 + 4n^3/3 = 2\left(n^2 m - n^3/3\right)
\end{aligned}$$

a cui vanno aggiunte $mn - n^2/2$ radici quadrate.

Rispetto all'algoritmo precedente, il metodo di Givens risulta essere meno complesso in termini di flops, ma può aver bisogno di un numero superiore di calcoli di radici quadrate (dipende dal valore di m e n).
L'implementazione di questo algoritmo su FPGA presenta tuttavia un notevole vantaggio, in quanto le operazioni svolte ad ogni iterazione dell'algoritmo (cancellazione di un elemento della matrice A) possono essere effettuate applicando l'algoritmo CORDIC (COordinate Rotation DIgital Computer) in due diverse modalità di funzionamento, come mostrato in [28].
Il CORDIC è un algoritmo iterativo mediante il quale è possibile ruotare vettori eseguendo soltanto operazioni elementari, ovvero somme (algebriche) e shift, per cui è facilmente descrivibile in HDL e si adatta bene ad implementazioni su FPGA. Esso può essere impiegato in due modalità: vettoriale e rotazionale.
In modalità vettoriale il CORDIC ruota il vettore di ingresso in modo da allinearlo con l'asse delle ascisse tendendo a minimizzare la componente verticale ad ogni iterazione; l'algoritmo converge dopo un certo numero di iterazioni fornendo come risultato l'angolo della rotazione e il modulo del vettore.
In modalità rotazionale il CORDIC ruota il vettore di ingresso di un certo angolo di rotazione (θ) fornito in ingresso; anche in questo caso l'algoritmo converge in un numero finito di iterazioni fornendo come risultato il vettore ruotato.
Poiché la fattorizzazione QR basata sull'algoritmo di Givens richiede un passo in cui si calcola l'angolo di rotazione di Givens (θ) necessario ad annullare l'elemento di A desiderato e un passo in cui questa rotazione viene applicata alla matrice A (descritta dalla relazione 3.22), l'algoritmo CORDIC si presta bene a svolgere queste due operazioni: utilizzando il CORDIC in modalità vettoriale si effettua il calcolo del valore dell'angolo di Givens e successivamente, utilizzando il CORDIC in modalità rotazionale, si effettua la rotazione.
Dunque grazie all'impiego del CORDIC l'implementazione su FPGA risulta vantaggiosa per le ragioni precedentemente esposte. Inoltre molti tool di

sintesi HDL (come ad esempio DSP builder) spesso forniscono tra le IP di libreria un blocco CORDIC che può funzionare nelle due modalità vettoriale e rotazionale: il flusso di progetto in questo caso risulta pertanto semplice e rapido.
Esempi di architetture basate sull'algoritmo di Givens sono [28, 29, 30].

3.6.4 Fattorizzazione Cholesky (LL^T)

L'algoritmo Cholesky fattorizza una matrice $A \in \mathbb{R}^{n \times n}$ semidefinita positiva nel prodotto di due matrici triangolari tali che:

$$A = G\,G^{\mathrm{T}}$$

La matrice G può essere triangolare inferiore o superiore; supponendo di considerare una matrice triangolare inferiore L, A sarà fattorizzata in:

$$A = L\,L^{\mathrm{T}}$$

Effettuando il prodotto tra L e L^{T} ed uguagliando gli elementi del prodotto a quelli di A si ottengono le seguenti relazioni:

$$\begin{aligned} l_{jj} &= \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2} \\ l_{ij} &= \sqrt{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}} \qquad \text{per } i > j \end{aligned} \tag{3.23}$$

Tuttavia è possibile pensare ad un modo più semplice per ricavare gli elementi di L; considerando j tale che $1 \leq j \leq n$, si ha[1]

$$a_j = \sum_{k=1}^{j} l_{jk}\, l_k$$

[1] nelle relazioni presentate gli elementi di matrici con un solo pedice rappresentano una colonna della matrice (a_j, l_k), mentre quelli con due pedici rappresentano un singolo elemento della matrice (l_{jj}, l_{jk}); gli elementi di vettori sono invece indicati con un solo pedice (v_j).

da cui:

$$l_{jj}\, l_j = a_j - \sum_{k=1}^{j-1} l_{jk}\, l_k \equiv v \in \mathbb{R}^n$$

per cui si può scrivere

$$l_j = v/l_{jj} = v/\sqrt{v_j}$$

tradotto in codice, l'algoritmo risulta:

for $j = 1 : n$
    $v(j : n) = A(j : n, j)$
    for $k = 1 : (j - 1)$
        $v(j : n) = v(j : n) - L(j, k)\, L(j : n, k)$
    end
    $L(j : n, j) = v(j : n)/\sqrt{v(j)}$
end

Esso può essere modificato in modo tale che la matrice L sovrascriva la matrice A per tutti gli $i \geq j$:

for $j = 1 : n$
    if $j > 1$
        $A(j : n, j) = A(j : n, j) - A(j : n, 1 : j - 1)\, A(j, 1 : j - 1)^{\mathrm{T}}$
    end
    $A(j : n, j) = A(j : n, j)/\sqrt{A(j, j)}$
end

Una volta ottenuta la fattorizzazione $L\, L^{\mathrm{T}}$ di A, si passa alla sostituzione diretta. L'equazione $Ax = L\, L^{\mathrm{T}} x = b$ può essere vista come:

$$\begin{cases} Ly = b \\ L^{\mathrm{T}} x = y \end{cases}$$

La sostituzione diretta consiste nel risolvere in y il sistema $Ly = b$. Essendo L una matrice triangolare inferiore la risoluzione è

$y(1) = b(1)/L(1,1)$
for $i = 2 : n$
    $y(i) = (b(i) - L(i, 1 : i-1)\, y(1 : i-1))/L(i,i)$
end

Si passa poi alla sostituzione inversa che consiste nella risoluzione del sistema $L^{\mathrm{T}}x = y$. Essendo L^{T} una matrice triangolare superiore, si ha

$x(n) = y(n)/L(n,n)$
for $i = (n-1) : -1 : 1$
    $x(i) = (y(i) - L(i, i+1 : n)\, x(i+1 : n))/L(i,i)$
end

Viene di seguito riportato il calcolo dei flops dell'algoritmo di fattorizzazione Cholesky: per il calcolo di $A(j : n, j)$ all'interno dell'**if** sono necessari $2(n - j)(j-1)$ flops per ogni iterazione del ciclo, mentre per il calcolo di $A(j : n, j)$ fuori dall'**if** sono necessari $(n - j)$ flops ed una radice quadrata per ogni iterazione del ciclo. I flops totali pertanto risultano:

$$\begin{aligned}
\text{flops} &= \sum_{j=2}^{n} [2(n-j)(j-1)] + \sum_{j=1}^{n}(n-j) = \\
&= \sum_{j=2}^{n}(jn - j^2 - n + j) + \sum_{j=1}^{n} n - \sum_{j=1}^{n} j \approx \\
&\approx 2\left(n\frac{n^2}{2} - \frac{n^3}{3} - n^2 + \frac{n^2}{2}\right) + n^2 - \frac{n^2}{2} = \\
&= 2\left(\frac{n^3}{6} - \frac{n^2}{2}\right) + \frac{n^2}{2} \approx \frac{n^3}{3}
\end{aligned}$$

a cui va aggiunto il calcolo di n radici quadrate. Quindi, per quanto riguarda l'algoritmo di fattorizzazione, esso risulta più semplice, in termini di flops, dei precedenti. Non va però dimenticato che è necessario effettuare dei prodotti matriciali prima di effettuare la fattorizzazione (come descritto dalla 3.19), per cui sono necessari n^2m flops per il calcolo di $B^{\mathrm{T}}B$ e nm flops per il calcolo di $B^{\mathrm{T}}y$. Inoltre anche i passaggi seguenti di sostituzione diretta e inversa dei metodi con fattorizzazione QR e con fattorizzazione Cholesky

hanno complessità diverse in termini di flops.
In conclusione la fattorizzazione QR con Givens rotation e la fattorizzazione Cholesky sono circa equivalenti: a seconda dell'applicazione e delle dimensioni delle matrici di dati può risultare più conveniente un algoritmo piuttosto che l'altro.
In [24] è riportato un esempio di architettura basata su questo algoritmo.

3.6.5 Fattorizzazione Cholesky modificata (LDL$^{\mathrm{T}}$)

Questo metodo consente di semplificare l'algoritmo precedente eliminando il calcolo delle radici quadrate precedentemente descritto così da ridurre il tempo di latenza e da semplificare l'architettura hardware del sistema.
Con questo metodo, la matrice A è fattorizzata come:

$$A = L\,D\,L^{\mathrm{T}}$$

dove

- D è una matrice diagonale;
- L è una matrice triangolare inferiore con elementi della diagonale unitari.

Il seguente codice descrive l'algoritmo; esso determina una matrice L triangolare inferiore ed una matrice D diagonale tale che $A = L\,D\,L^{\mathrm{T}}$, sovrascrivendo l'elemento a_{ij} della matrice A con l_{ij} se $i > j$ e con d_{ii} se $i = j$:

for $j = 1 : n$
    for $i = 1 : (j-1)$
        $v(i) = A(j,i)\,A(i,i)$
    end
    $A(j,j) = A(j,j) - A(j,1:j-1)\,v(1:j-1)$
    $A(j+1:n,j) = (A(j+1:n,j) - A(j+1:n,1:j-1)\,v(1:j-1))\,/A(j,j)$
end

Si nota che in questo algoritmo non sono coinvolti calcoli di radici quadrate.
Calcolo dei flops: per il calcolo di $v(i)$ sono necessari $(j-1)$ flops per ogni

iterazione del ciclo **for** più esterno; il calcolo di $A(j,j)$ richiede $(2j+1)$ flops per ogni iterazione dello stesso **for**, mentre il calcolo di $A(j+1:n,j)$ ne richiede $2(n-j)j$. Dunque si ha:

$$\begin{aligned}\text{flops} &= \sum_{j=1}^{n}(j-1)+\sum_{j=1}^{n}(2j+1)+\sum_{j=1}^{n}2(n-j)j = \\ &= \frac{n}{2}(n-1)-n+n+2\sum_{j=1}^{n}j+2\sum_{j=1}^{n}(n-j)j \approx \\ &\approx \frac{n^2}{2}+2\frac{n^2}{2}+2\sum_{j=1}^{n}(nj-j^2) = \\ &= \frac{3n^2}{2}+2\left(\sum_{j=1}^{n}j+\sum_{j=1}^{n}j^2\right) \approx 2\left(\frac{n^3}{2}-\frac{n^3}{3}\right)=\frac{n^3}{3}\end{aligned}$$

La complessità dell'algoritmo in termini di flops è dunque equivalente a quella dell'algoritmo Cholesky con la differenza che adesso non è necessario svolgere il calcolo delle radici quadrate.
Una volta ottenuta la fattorizzazione $\mathrm{LDL}^{\mathrm{T}}$ di A, per ottenere la soluzione del problema LS si passa alle sostituzioni, che in questo caso sono tre. L'equazione $L\,D\,L^{\mathrm{T}}\,x=b$ può infatti essere vista come:

$$\begin{cases} L\,z=b \\ D\,h=z \\ L^{\mathrm{T}}x=h \end{cases}$$

1. Risoluzione in z di $L\,z=b$:

```
z(1) = b(1)
for i = 2 : n
    z(i) = b(i) − L(i, 1 : i − 1) z(1 : i − 1)
end
```

2. Risoluzione in h di $D\,h=z$

```
for i = 1 : n
    h(i) = z(i)/d(i)
end
```

3. Risoluzione in x di $L^{\mathrm{T}}x = h$

$x(n) = h(n)$
for $i = n-1 : -1 : 1$
    $x(i) = h(i) - L(i, i+1:n)\, x(i+1:n)$
end

Si nota che le operazioni di divisione sono presenti soltanto nel passo 2; ciò costituisce un vantaggio rispetto alla soluzione precedente, dove il tempo di latenza dovuto alla divisione presente nel primo dei due passi riduceva le prestazioni del sistema; infatti adesso le operazioni di divisione sono presenti in un passo separato (il passo 2 appunto) per cui si può aggiungere uno stadio di pipeline così da aumentare il throughput ed incrementare le prestazioni del sistema. Ciò rappresenta un secondo vantaggio di questa soluzione rispetto alla precedente.
In [31] è presentata un'architettura che fa uso di questo metodo per l'implementazione di tale algoritmo su FPGA.

Un confronto tra le caratteristiche principali degli algoritmi di risoluzione dei problemi dei minimi quadrati presentati è riportato in tabella 3.1.
Si può dunque concludere che due tra gli algoritmi presentati, all'incirca equivalenti tra loro in termini di complessità, sono risultati essere più vantaggiosi rispetto agli altri: la fattorizzazione QR con l'algoritmo di Givens e la fattorizzazione $\mathrm{LDL}^{\mathrm{T}}$. Entrambi si prestano bene ad essere implementati su FPGA; la scelta di uno o dell'altro dipenderà dall'applicazione e dalle dimensioni delle matrici di dati su cui si dovrà applicare l'algoritmo.

algoritmo	caratteristiche
fatt. QR (Grahm Schmidt)	algoritmo complesso in termini di flops, richiede il calcolo di n radici quadrate
fatt. QR (Givens rotation)	algoritmo meno complesso, possibilità di utilizzare l'algoritmo CORDIC per risolvere i calcoli delle radici quadrate, adatto ad una implementazione su FPGA
fatt. Cholesky ($\mathbf{LL^T}$)	complessità circa equivalente a fatt. QR con algoritmo di Givens, richiede il calcolo di n radici quadrate
fatt. Cholesky modificata ($\mathbf{LDL^T}$)	complessità circa equivalente a fatt. QR con algoritmo di Givens, non richiede calcoli di radici quadrate, adatto ad una implementazione su FPGA

Tabella 3.1

Capitolo 4

Implementazione dell'algoritmo di stima su FPGA

In questo capitolo sono descritti e discussi i dettagli implementativi relativi alla realizzazione su FPGA dell'algoritmo di stima del SoC descritto nel paragrafo precedente. Dopo una breve descrizione dell'hardware e dei CAD impiegati, si propone una possibile architettura di quella sezione del BMS dedicata alla stima del SoC con particolare riferimento al partizionamento hardware-software. Infine viene descritto nel dettaglio il sistema realizzato.

4.1 Scelta dell'FPGA e dei CAD di sviluppo

Per la realizzazione dell'algoritmo di stima del SoC si è scelto di utilizzare FPGA e tools di sviluppo Altera; in particolare si è scelto di utilizzare la scheda di sviluppo Altera DE0-Nano, una piattaforma di sviluppo FPGA compatta adatta a numerose applicazioni. Nel caso in questione si adatta bene alle necessità di progetto in quanto possiede un Altera Cyclone IV FPGA (con 22320 elementi logici), buone risorse di memoria (32 MB di memoria SDRAM, 2 Kb di memoria EEPROM) e ha integrato un convertitore A/D della National Semiconductor a 12 bit e 8 canali, necessario per l'acquisizione delle variabili di ingresso del sistema (ovvero tensione di cella e corrente di carico).
L'FPGA presente su tale scheda di sviluppo è adatto ad essere utilizzato

per realizzare sistemi embedded con processori soft-core, i quali consentono di ottenere una elevata flessibilità grazie alla possibilità di effettuare un partizionamento del sistema in una componente hardware ed una una componente software: si possono così suddividere le funzionalità del sistema in una componente svolta interamente via software dal processore integrato ed in una componente hardware realizzata tramite le risorse logiche dell'FPGA. Entrambe queste componenti possono operare in parallelo, consentendo di ottenere ottime prestazioni.
Per la configurazione e la programmazione dell'FPGA è stato utilizzato il software Quartus II di Altera, e come processore soft-core si è utilizzato il Nios II.

Tipicamente il flusso di progetto di un sistema embedded realizzato su FPGA, superata la fase di validazione dell'algoritmo da implementare, prevede di effettuare il partizionamento hardware-software e successivamente di sviluppare il software nel linguaggio di programmazione scelto (solitamente il C) e di descrivere la parte hardware in HDL (Verilog o VHDL). Viene dunque effettuata una suddivisione del flusso di progetto in due flussi paralleli, un flusso hardware ed un flusso software. In questo lavoro di tesi è stato scelto di utilizzare un flusso di progettazione hardware diverso ed innovativo, che fa uso del tool DSP builder di Altera.
DSP builder è un tool che opera su Simulink, in ambiente MATLAB, che, aggiungendo delle librerie specifiche a quelle di Simulink, consente di implementare sistemi di Digital Signal Processing in maniera semplice e veloce. Esso infatti consente di creare una rappresentazione del sistema in un ambiente favorevole allo sviluppo di algoritmi, quale l'ambiente Simulink: sono infatti integrate le funzionalità di sviluppo dell'algoritmo, simulazione e verifica tipiche di MATLAB e Simulink con il software Quartus II per la configurazione dell'FPGA. DSP Builder infatti realizza una sintesi hardware che traduce l'algoritmo sviluppato e simulato ad alto livello in Simulink in un hardware a basso livello ottimizzato per l'FPGA (Altera) specificato e per la frequenza di clock selezionata. Tale sintesi è espressa in codice VHDL, che può essere utilizzato da Quartus II per generare uno o più moduli hardware

da interfacciare al processore.
Esistono due versioni di DSP builder: Standard Blockset e Advanced Blockset. Lo Standard Blockset consente di realizzare la operazioni aritmetiche di base e funzioni di memorizzazione, oltre alla possibilità di generare dispositivi essenziali come PLLs, blocchi DSP, embedded memory. E' possibile inoltre aggiungere delle funzioni complesse includendo delle IP nel progetto; tali IP possono essere anche generate dall'utente in quanto è possibile importare blocchi descritti in HDL da un progetto Quartus II. L'Advanced Blockset, rispetto allo Standard, presenta i seguenti vantaggi:

- Pipelining automatico per far rispettare i time constraints anche a frequenze elevate di clock di 300-400 MHz
- Condivisione di risorse automatica
- Possibilità di realizzare progetti in virgola mobile ad alte prestazioni

Avendo a disposizione entrambe le librerie, si è scelto di utilizzare l'Advanced Blockset per i vantaggi sopra elencati: in particolare, la possibilità di realizzare progetti interamente in virgola mobile con precisione definibile dall'utente (singola o doppia) in maniera semplice e veloce costituisce uno strumento di progettazione molto potente. Gli FPGA infatti non costituiscono la scelta più "classica" per applicazioni in virgola mobile; sebbene spesso i venditori forniscano numerose librerie in proposito, le prestazioni degli FPGA per applicazioni floating point sono spesso limitate, soprattutto a causa delle caratteristiche degli operatori in virgola mobile, i quali sono caratterizzati da strutture complesse e molti stadi di pipeline che generano una elevata latenza e congestione del routing. Per questo motivo il progetto finale spesso ha una frequenza massima di funzionamento bassa.
La sintesi generata dall'Advanced Blockset di DSP builder, invece, unisce le varie parti del datapath in un'unica funzione in virgola mobile ottimizzata per l'operazione da svolgere e per il tipo di input, che consente un risparmio in termini di risorse e di latenza [23]. Dunque la scelta dell'Advanced Blockset ha consentito di sfruttare i vantaggi dell'utilizzo della virgola mobile in modo semplice e rapido.

4.2 Flusso di progetto

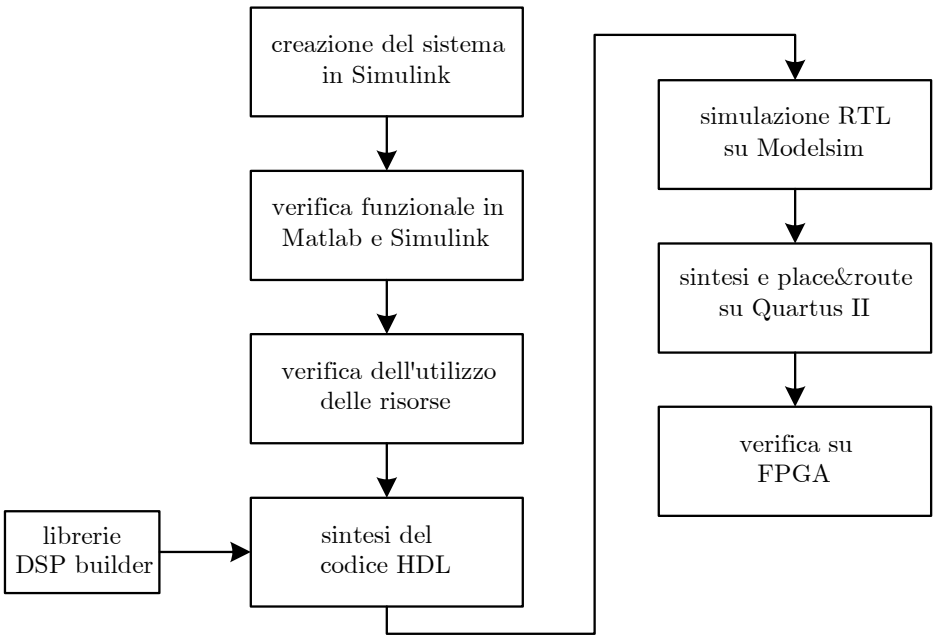

Figura 4.1: Schema del flusso di progetto

Vediamo dunque com'è strutturato il flusso di progetto. La Figura 4.1 mostra i vari passi del flusso di progetto; come si può notare, sono presenti più fasi di verifica, sia a livello simulativo sia a livello pratico. Esso è descritto dai seguenti passi:

1. Si crea un modello in DSP builder dell'algoritmo da implementare;
2. Si effettua una verifica della funzionalità dell'algoritmo a livello simulativo in Simulink e MATLAB. Se necessario, si modifica il modello dell'algoritmo precedentemente generato. La compilazione da parte di DSP builder genera due elementi: il codice VHDL del sistema, strutturato in maniera tale che la top-level entity e le entity sottostanti coincidano con la struttura dei sotto-sistemi di cui è costituito il sistema, e i testbench per la verifica RTL su Modelsim.
3. Si effettua una verifica dell'utilizzo delle risorse; questa verifica non avviene a livello hardware, ma è semplicemente svolta tramite una stima

dell'impiego delle risorse (stima generata da DSP builder). Ciò consente già a questo livello di attuare una ottimizzazione delle risorse, resa possibile dalla modifica di alcuni parametri direttamente in DSP builder.

4. Si effettua una verifica funzionale e timing in Modelsim; a seconda dell'esito può essere necessario modificare il sistema in DSP builder oppure direttamente il VHDL.
5. Si include il codice HDL in Quartus II e si avvia la compilazione. Se ha esito positivo si passa alla verifica diretta su FPGA.

4.3 Partizionamento hardware-software

Presa coscienza delle potenzialità offerte dai CAD di sviluppo a disposizione e dell'efficacia del flusso di progetto appena descritto, è possibile pensare di poter realizzare un BMS per una generica batteria al litio come sistema embedded interamente integrato su FPGA, a condizione di scegliere un FPGA con sufficienti risorse (come vedremo l'FPGA a disposizione per questo lavoro di tesi non è sufficiente).
A tale scopo è necessario sviluppare un opportuno partizionamento hardware-software che consenta di ottenere un sistema da un lato affidabile e robusto, e dall'altro lato caratterizzato da buone prestazioni in termini di tempi di calcolo; tutte queste caratteristiche devono essere tali da rendere il sistema competitivo in termini di prestazioni e costi rispetto ad una realizzazione interamente software attuabile su processori general purpose o su DSPs.
In questo lavoro di tesi si propone di realizzare un partizionamento hardware-software che consente di svincolare completamente il processo di stima del SoC di ciascuna cella della batteria dal software, rendendo tale processo del tutto parallelo alle operazioni svolte dal processore. Ciò è possibile se la stima del SoC viene effettuata interamente in un blocco di accelerazione hardware (*hardware accelerator*) a cui il software possa accedere qualora ne abbia bisogno: dunque la parte di software risulta indipendente dal processo di stima che altrimenti risulterebbe oneroso in termini di tempi di calcolo e di risorse, e può quindi essere dedicata allo svolgimento delle altre funzioni del BMS.
Il partizionamento proposto è rappresentato in Figura 4.2: esso è costituito

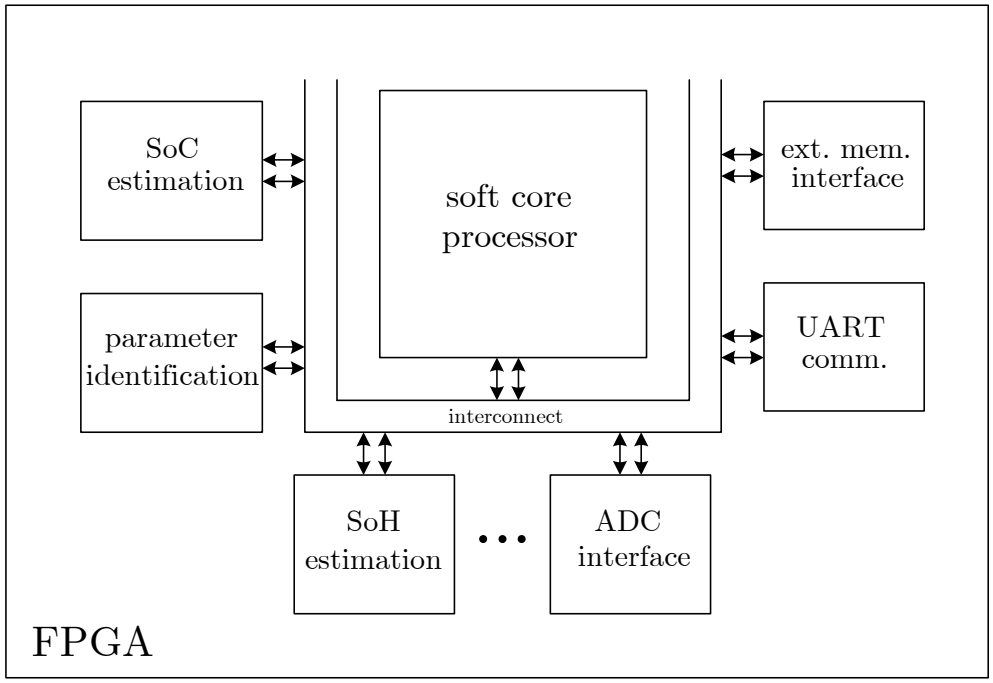

Figura 4.2: Possibile architettura FPGA del sistema di stima del SoC di un BMS

da un modulo di stima del SoC che può essere impiegato per la stima del SoC di tutte le celle della batteria, un modulo di identificazione dei parametri, anch'esso utilizzabile per l'identificazione dei parametri di tutte le celle, che aggiorna i parametri del modello circuitale contenuti nel modulo di stima del SoC, e un modulo per la stima del SoH della batteria. Dato che per un regime di utilizzo "usuale" della batteria il valore del SoC varia lentamente e dato che la frequenza con cui la stima del SoC e l'identificazione dei parametri devono essere calcolate (al massimo qualche hertz) è molto inferiore alla frequenza di clock del sistema (tipicamente dell'ordine del centinaio di megahertz), i due moduli di stima del SoC e di identificazione dei parametri possono essere utilizzati in time sharing per tutte le celle, senza bisogno di utilizzare un modulo hardware per ogni cella. Il compito del software nei confronti di questi moduli è semplicemente quello di configurazione e di lettura dei dati, tutto il resto avviene in maniera indipendente: anche il flusso dei dati provenienti dall'ADC può essere reso indipendente dal software con

l'aggiunta di un DMA dedicato.

In questo lavoro di tesi è stato studiato e implementato il modulo di stima del SoC. Il modulo hardware da inserire nello schema di Figura 4.2 è rappresentato in Figura 4.3. Esso deve avere come ingressi, oltre all'enable,

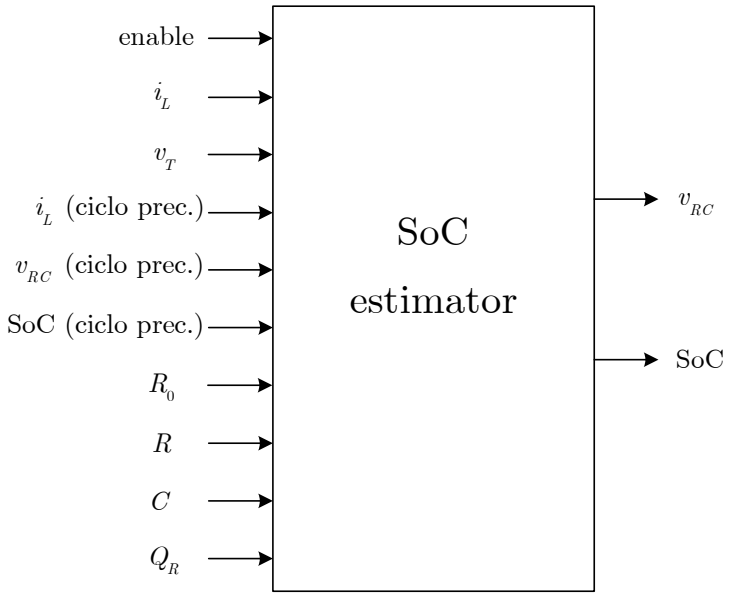

Figura 4.3: Modulo hardware di stima del SoC

il valore campionato delle due grandezze di ingresso i_L e v_T, i valori di R_0, R e C del modello provenienti dal modulo di identificazione dei parametri, il valore di Q_R proveniente dal modulo di stima del SoH, e, poiché tale modulo è condiviso tra tutte le celle, sono necessari anche come ingressi i valori, calcolati al passo di stima del SoC precedente, delle variabili che durante il calcolo vengono ritardate di un ciclo (ovvero moltiplicate per z^{-1}): i_L, v_R e SoC. Come uscite il modulo deve avere il valore del SoC al ciclo attuale e il valore di v_{RC} al ciclo attuale, che nel passo successivo di stima verrà inviato all'ingresso di v_{RC} relativo al ciclo precedente.
Tutti gli ingressi e le uscite del modulo possono semplicemente essere memorizzate in un set di registri, uno per ogni cella: al momento di effettuare il passaggio di calcolo del SoC da una cella ad un'altra basta effettuare uno

scambio di dati nei registri. E' importante notare che con una soluzione di questo tipo si suppone che la curva SoC-OCV sia la stessa per tutte le celle (infatti una LUT per ogni cella comporterebbe un elevato consumo di risorse); tale ipotesi trova conferma in letteratura.

4.4 Realizzazione del sistema in DSP builder

In questo lavoro di tesi è stato realizzato il modulo di stima del SoC, descritto in precedenza, ed è stato testato utilizzando una cella di batteria al litio di tipo NMC. Dovendo tale modulo operare su un'unica cella, la sua architettura differisce leggermente da quella descritta nel precedente paragrafo; in particolare esso non ha gli ingressi di i_L, v_R e SoC relativi al ciclo precedente, né l'uscita v_{RC} relativa al ciclo corrente in quanto non c'è bisogno di condividere gli elementi di calcolo con dati provenienti da altre celle. Dunque i ritardi di un ciclo del processo di stima sono in questo caso realizzati con un semplice blocco di ritardo (*sample delay*).
Inoltre, rispetto al modulo precedentemente descritto, il modulo di stima del SoC realizzato ha altri due ingressi: un ingresso per il periodo di campionamento T dei dati in ingresso, e un ingresso per il valore di inizializzazione del SoC, ovvero $\mathrm{SoC}_{\mathrm{init}}$. Come uscita, oltre al valore del SoC stimato, si è scelto di aggiungere anche la tensione v_M di uscita del modello per poter osservare il suo andamento e confrontarlo con la tensione di cella.
In Figura 4.4 è mostrata la top level entity del modulo di stima del SoC realizzato; la Figura 4.5, invece, mostra la struttura del sistema nel livello gerarchico inferiore. Si notano bene tutti i vari blocchi che costituiscono lo schema di Figura 3.3: un blocco di Coulomb Counting per il calcolo del SoC a partire dai campioni di i_L, un blocco per il calcolo della tensione v_{RC} ed uno per il calcolo della tensione v_{R_0}, una LUT contenente la caratteristica SoC-OCV misurata tramite test offline e un blocco per il calcolo del guadagno di retroazione K_P.

Il blocco di Coulomb Counting svolge l'operazione di integrazione della carica determinando il SoC normalizzato a Q_R a partire da un valore di SoC

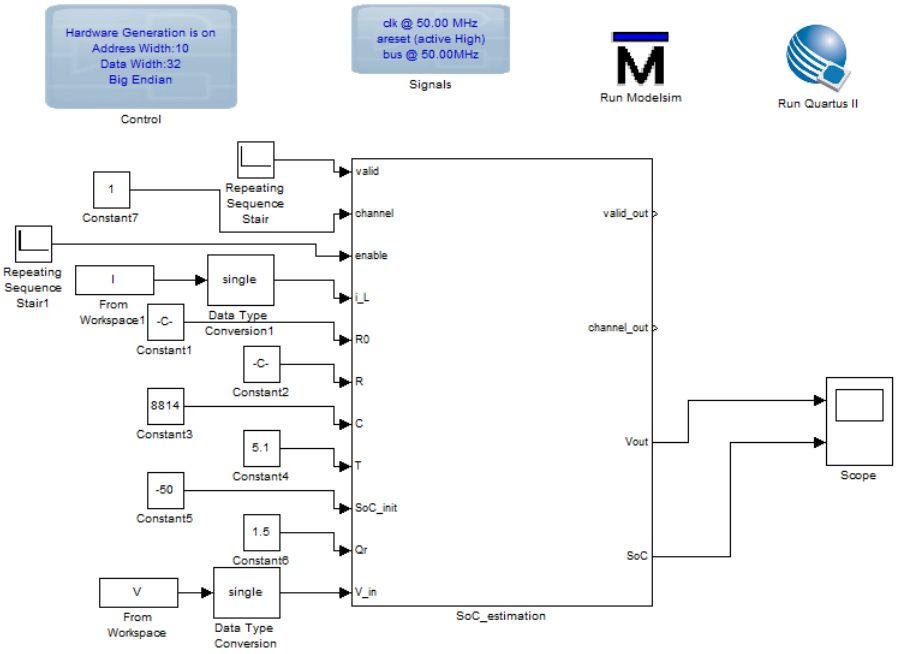

Figura 4.4: Top level entity del modulo di stima del SoC

iniziale pari a $\mathrm{SoC}_{\mathrm{init}}$. L'operazione che svolge è dunque descritta da:

$$\mathrm{SoC}(t) = \mathrm{SoC}_{\mathrm{init}} - \frac{1}{Q_R}\int_{t_0}^{t} i_L(\tau)\,\mathrm{d}\tau$$

applicando a questa relazione la trasformata di Laplace si ha:

$$\mathrm{SoC}(s) = -\frac{I_L(s)}{Q_R s}$$

utilizzando la trasformazione di Tustin si passa al dominio discreto:

$$\mathrm{SoC}(z) = -\frac{I_L(z)}{Q_R}\,\frac{T}{2}\,\frac{1+z^{-1}}{1-z^{-1}}$$

$$\mathrm{SoC}(z) = \mathrm{SoC}(z)\cdot z^{-1} - I_L(z)\,\frac{T}{2Q_R}\,(1+z^{-1}) \tag{4.1}$$

Poiché la moltiplicazione di un segnale nel dominio discreto per $z^{-}1$ corrisponde a ritardare il segnale di un periodo di campionamento, si ha che il

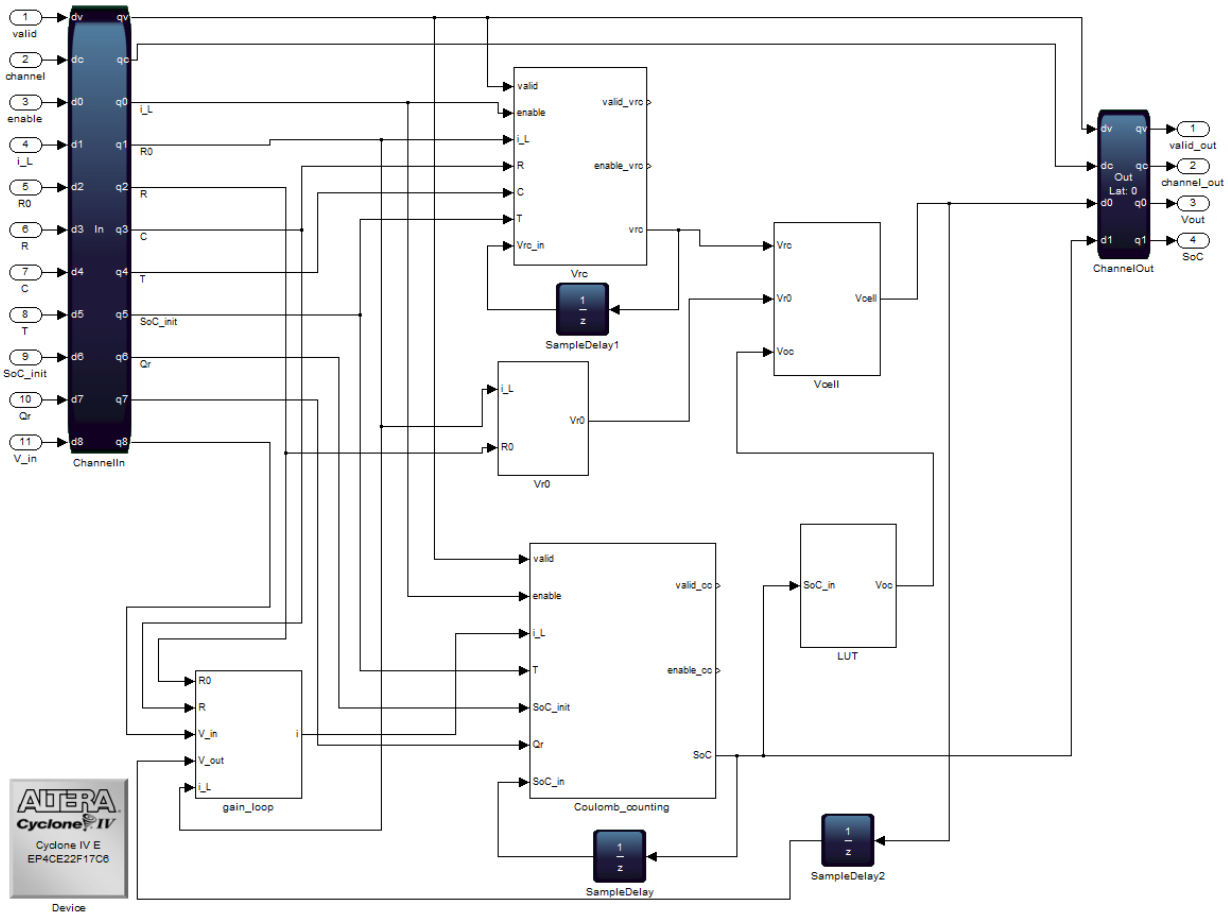

Figura 4.5: Modulo di stima del SoC

SoC all'istante attuale dipende sia dal SoC all'istante precedente che dalla corrente all'istante attuale e all'istante precedente (la cui somma costituisce infatti un passo di integrazione). Risulta evidente quindi perché nel modulo di stima del SoC che può essere condiviso tra più celle descritto nel precedente paragrafo fosse necessario conoscere il valore di SoC e di i_L all'istante di campionamento precedente.

In Figura 4.6 è riportato il blocco di Coulomb Counting realizzato su Simulink; si nota che la 4.1 è stata realizzata utilizzando blocchi di somma, moltiplicazione, divisione e ritardo appartenenti alla libreria ModelPrim di DSP builder. Si nota inoltre che il blocco di ritardo possiede un enable in quanto esso deve essere attuato soltanto una volta ogni periodo di campiona-

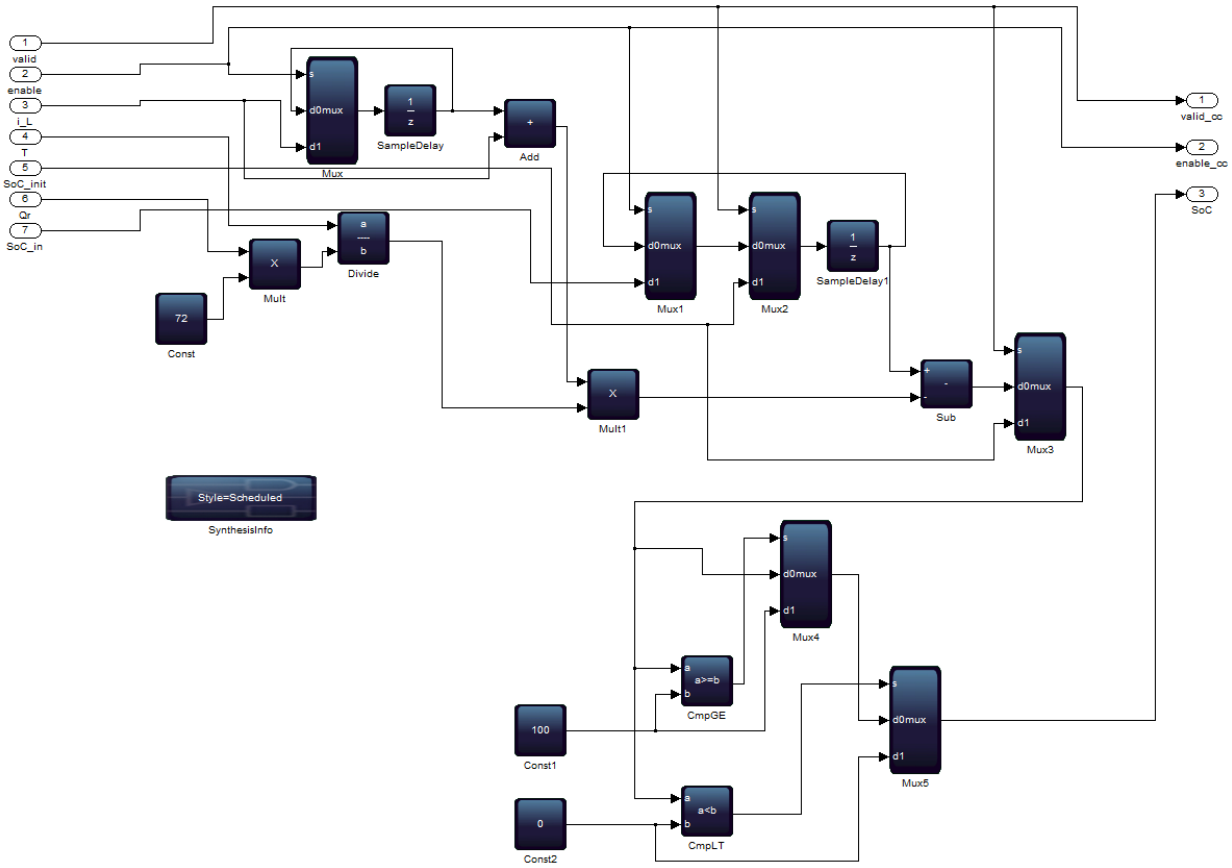

Figura 4.6: Modulo di Coulomb Counting

mento dei segnali (frequenza di 10 Hz nel caso particolare) e non una volta ogni ciclo di clock (frequenza di 50 MHz). Infine si nota la presenza di una serie di elementi logici la cui funzione è quella di limitare il SoC (che in questa implementazione è calcolato in percentuale) ad un range di valori compresi tra 0 e 100.

Il blocco per il calcolo della v_{RC} determina la tensione ai capi del parallelo tra R e C; essa è data da:

$$\frac{\mathrm{d}}{\mathrm{d}t}v_{RC}(t) = -\frac{v_{RC}(t)}{RC} + \frac{i_L(t)}{C}$$

applicando la trasformata di Laplace:

$$s \cdot V_{RC}(s) = -\frac{1}{C}\left(\frac{V_{RC}(s)}{R} + I_L(s)\right)$$

e passando al dominio discreto:

$$\begin{aligned}
V_{RC}(z)\,\frac{2}{T}\,\frac{1-z^{-1}}{1+z^{-1}} &= -\frac{1}{C}\left(\frac{V_{RC}(z)}{R} + I_L(z)\right) \\
V_{RC}(z) &= -\frac{T}{2RC}\,V_{RC}(z)\,\frac{1+z^{-1}}{1-z^{-1}} - \frac{T}{2C}\,I_L(z)\,\frac{1+z^{-1}}{1-z^{-1}} \\
V_{RC}(z)\left(1 + \frac{T}{2RC}\,\frac{1+z^{-1}}{1-z^{-1}}\right) &= -\frac{T}{2C}\,I_L(z)\,\frac{1+z^{-1}}{1-z^{-1}}
\end{aligned}$$

separando poi i termini all'istante attuale da quelli all'istante di campionamento precedente (ovvero moltiplicati per z^{-1}) si ottiene:

$$\begin{aligned}
V_{RC}(z) - z^{-1}\cdot V_{RC}(z) + \frac{T}{2RC}V_{RC}(z) + \frac{T}{2RC}\,V_{RC}(z)\cdot z^{-1} &= -\frac{T}{2C}\,I_L(z)\,\frac{1+z^{-1}}{1-z^{-1}} \\
V_{RC}(z)\left(\frac{2RC+T}{2RC}\right) - z^{-1}\cdot V_{RC}(z)\left(\frac{2RC-T}{2RC}\right) &= -\frac{T}{2C}\,I_L(z)\,\frac{1+z^{-1}}{1-z^{-1}}
\end{aligned}$$

da cui

$$V_{RC}(z) = z^{-1}\cdot V_{RC}(z)\left(\frac{2RC-T}{2RC+T}\right) - \frac{RT}{2RC+T}\,I_L(z)\,(1+z^{-1}) \qquad (4.2)$$

Si nota dunque che $V_{RC}(z)$ all'istante attuale dipende, oltre che dal campione attuale di i_L anche dai campioni all'istante di campionamento precedente di i_L e v_{RC}. Ciò giustifica la presenza dell'ingresso v_{RC} al ciclo precedente nel modulo di stima del SoC di Figura 4.3.

La Figura 4.3 mostra la realizzazione in Simulink del blocco per il calcolo della v_{RC}; si nota che la 4.2 è stata anche in questo caso realizzata utilizzando blocchi di somma, sottrazione, moltiplicazione, divisione e ritardo (con enable).

Gli altri blocchi sono piuttosto semplici. Il blocco per il calcolo della v_{R_0} è costituito semplicemente da un moltiplicatore che effettua la moltiplicazione

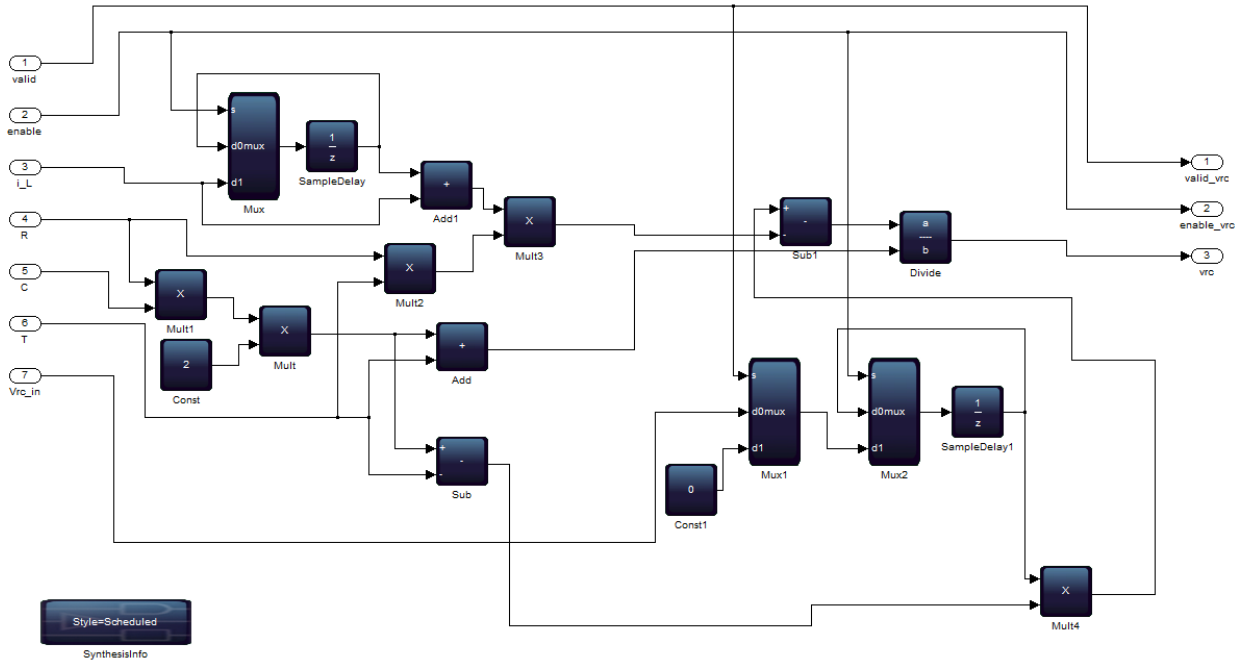

Figura 4.7: Blocco di calcolo della v_{RC}

tra R_0 e i_L. Il blocco di LUT invece è costituito da una look-up table contenente i valori di OCV indirizzata dal valore del SoC determinato dal blocco di Coulomb Counting; tali punti sono stati determinati tramite un test che verrà descritto nel capitolo successivo. Il blocco di gain determina il valore di $K_{P_{\text{opt}}}$ come $1/(R_0 + R)$ e lo moltiplica alla differenza tra v_M e v_T, e sottrae poi il risultato da i_L. Infine il blocco di calcolo della tensione di cella effettua una sottrazione tra la tensione V_{OC} e le tensioni v_{RC} e v_{R_0} per determinare la tensione di uscita del modello v_M.

4.5 Impiego delle risorse dell'FPGA

Un aspetto importante della progettazione del sistema è l'impiego delle risorse dell'FPGA, in quanto trascurare questo aspetto potrebbe comportare il consumo completo delle risorse disponibili, le quali sono limitate. Nel caso dell'FPGA impiegato le risorse disponibili sono: 22320 elementi logici totali, 154 pins, 608256 memory bits, 132 embedded multipliers (9-bits), 4 PLLs. Per il sistema realizzato il consumo di risorse coinvolge in maniera più critica

gli elementi logici totali e i moltiplicatori integrati.
Poiché nel flusso di progetto adottato la sintesi dell'hardware avviene in maniera automatica tramite DSP builder, non è semplice predire il consumo di risorse, per cui è utile utilizzare lo strumento di stima delle risorse (resource usage) di DSP builder e cercare di ottimizzare il progetto in modo tale da impiegare meno risorse possibile. Il compilatore floating-point di Altera, infatti, fonde una parte del flusso di dati generato da più operazioni in virgola mobile in un'unica funzione invece che costruirlo a partire dalla composizione di più operatori. Ciò è realizzato scegliendo la normalizzazione ottima degli ingressi in modo tale da eliminare più passi di normalizzazione/de-normalizzazione possibile, i quali risulterebbero invece presenti nell'unione di più operatori in virgola mobile [23]. Dunque il consumo di risorse dipende dal datapath del sistema.

Nel sistema implementato il consumo di risorse risulta piuttosto elevato

Flow Summary	
Flow Status	Successful - Mon Jan 13 09:39:40 2014
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	cell_model_SoC_estimation
Top-level Entity Name	DE0_Nano_Basic_Computer
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	20,455 / 22,320 (92 %)
Total combinational functions	14,413 / 22,320 (65 %)
Dedicated logic registers	15,066 / 22,320 (68 %)
Total registers	15135
Total pins	140 / 154 (91 %)
Total virtual pins	0
Total memory bits	146,720 / 608,256 (24 %)
Embedded Multiplier 9-bit elements	125 / 132 (95 %)
Total PLLs	1 / 4 (25 %)

Figura 4.8: Impiego delle risorse del sistema realizzato

a causa della presenza di sommatori in virgola mobile e di moltiplicatori/divisori, anch'essi in virgola mobile. Infatti ciascun sommatore (o sottrattore) impiega un numero di risorse pari a circa il 4-5% degli elementi logici totali, ciascun moltiplicatore circa il 3% degli elementi logici e 7 embedded-multipliers, e ciascun divisore poco meno del 6% degli elementi logici e ben 22 embedded multipliers. L'impiego dei moltiplicatori è critico in quanto qua-

lora finissero i moltiplicatori e i divisori verrebbero implementati sfruttando soltanto elementi logici, per cui il consumo di risorse aumenterebbe considerevolmente. È dunque importante sfruttare al meglio i 132 moltiplicatori integrati disponibili.
Il consumo di risorse complessivo del sistema realizzato è mostrato in Figura 4.8, la quale mostra il flow summary di Quartus II a compilazione terminata. Si nota che il consumo dei moltiplicatori e degli elementi logici totali ha quasi raggiunto il limite massimo, in quanto supera il 90% per entrambi. Risulta dunque chiaro che per la realizzazione del sistema comprensivo del modulo di identificazione dei parametri l'FPGA utilizzato non è sufficiente; occorre dunque utilizzare un FPGA con più risorse.
È possibile dunque fare una stima sull'impiego delle risorse logiche in funzione del numero di celle della batteria. Poiché in precedenza è stato mostrato che il modulo di stima del SoC realizzato, con poche modifiche, può essere utilizzato in time sharing per tutte le celle, le risorse logiche utilizzate per n celle saranno all'incirca le stesse che quelle utilizzate in questo lavoro, ovvero per una singola cella. Infatti le modifiche da apportare al modulo consistono nel rimuovere tre registri (con enable) interni, senza modificare il numero di moltiplicatori, divisori o sommatori/sottrattori, i quali costituiscono gli elementi di maggior consumo delle risorse logiche. L'unica componente del sistema che dipende dal numero di celle sono i set di registri che memorizzano gli ingressi/uscite del modulo di stima del SoC, in quanto occorre uno di questi set per ogni cella; comunque il consumo di risorse dovuto a tali set di registri è trascurabile rispetto al modulo di stima del SoC, per cui è ragionevole supporre che le risorse necessarie non aumentino molto al variare del numero di celle.

Capitolo 5

Risultati sperimentali

In questo capitolo vengono mostrati i risultati ottenuti dai test effettuati sulla cella. Dopo una breve introduzione sulle caratteristiche della cella e sul set di verifica impiegato, vengono presentati e valutati i risultati dei test eseguiti, confrontando tali risultati sia con valori di riferimento, sia con i valori ottenuti dalle simulazioni in Simulink.

5.1 Caratterizzazione della batteria

Prima di passare all'analisi dei risultati sperimentali ottenuti è necessario descrivere la cella di riferimento impiegata nello sviluppo del progetto e la sue caratteristiche ottenute da test di misura; in particolare questi test sono stati svolti per ottenere informazioni sulla caratteristica SoC-OCV e sui parametri del modello della cella necessari al testing del sistema di stima del SoC realizzato.

5.1.1 Cella di riferimento

Nello sviluppo di questa tesi è stata impiegata una cella di piccola capacità in modo da agevolare l'esecuzione dei test con strumentazione standard da laboratorio, senza perdere in generalità: la batteria SLPB723870H4, una cella NMC da 1.5 Ah della Kokam appartenente alla categoria delle batterie ultra high power. Le sue caratteristiche principali sono riportate in tabella. La densità di energia di cui dispone è di 150 watt-ora per Kg (Wh/Kg) e può

SPECIFICHE CELLA		
capacità nominale		1.5 Ah
tensione nominale		3.7 V
condizioni di carica	corrente massima	1.5 A
	tensione massima	4.2 V
condizioni di scarica	massima corrente continua	30.0 A
	massima corrente impulsata	60.0 A
	tensione di cut-off	2.7 V
cicli totali di scarica		> 500
temperatura operativa	carica	0 – 40 °C
	scarica	-20 – 60 °C

essere caricata e scaricata nell'intervallo di tensioni 2.7 V - 4.2 V con correnti massime che possono arrivare a 1C per la carica (ovvero 1.5 A; 1C è pari alla corrente necessaria a scaricare/caricare completamente la batteria in 1 ora) e 20C per la scarica (ovvero 30 A).
Il sistema di stima dello stato di carica sviluppato in questa tesi è pensato per una applicazione PHEV (Plug-in Hybrid Electric Vehicle), ovvero per veicoli ibridi nei quali, come sottolineato nel capitolo 2, la stima del SoC è particolarmente complessa a causa del fatto che la batteria non raggiunge mai uno stato di carica noto, come ad esempio lo stato di carica completa. L'algoritmo implementato, descritto nel capitolo precedente, consente una stima accurata del SoC anche in queste condizioni, e dunque si adatta bene ad una applicazione di questo tipo.

5.1.2 Caratteristica SoC-OCV

Come visto nei capitoli precedenti, la conoscenza della caratteristica SoC-OCV è essenziale per la stima del SoC. In particolare, nell'algoritmo implementato la curva SoC-OCV è necessaria per il calcolo della tensione di uscita del modello.
Dunque un primo passo per la realizzazione hardware del sistema di stima del SoC è il calcolo della caratteristica SoC-OCV, ottenuto elaborando i dati

acquisiti dai tests sulla cella. Tali tests sono stati realizzati effettuando dei cicli completi di carica e scarica della cella con l'ausilio di un Keithley 2420, una source-meter unit che funziona sia da sorgente/carico della cella, sia da strumento di misura di corrente e tensione.
Test di questo tipo, tuttavia, risultano tanto più lunghi quanto più aumenta l'accuratezza sulla misura della curva. Infatti la misura dell'OCV richiede un tempo di rilassamento che, nel caso della cella in questione, è dell'ordine di 30 minuti - 1 ora, tempo che impiega la tensione di cella (a corrente nulla in quanto il carico è stato scollegato) ad andare a regime; dunque per ciascuna misura dell'OCV è necessario attendere un tempo pari al tempo di rilassamento. E' chiaro che il numero di misure dell'OCV non può essere troppo elevato perché altrimenti il tempo di test risulterebbe eccessivamente lungo, per cui la curva SoC-OCV deve essere ricostruita per punti tramite un'interpolazione.

Nel test effettuato si è scelto di realizzare una scarica completa costituita da intervalli di scarica del 5% della capacità nominale Q_n (dunque ciascuna scarica a C/20), e da una carica completa della cella anch'essa con intervalli del 5%. Tra un intervallo e l'altro è presente un rilassamento della durata di 30 minuti, al termine del quale la tensione di cella può essere considerata "a regime". Inoltre, tra un ciclo completo e l'altro è presente una pausa di un'ora. L'andamento della tensione di cella e della corrente di carico nel tempo è riportato in Figura 5.1, nella quale si può notare l'andamento, approssimabile ad un esponenziale, della tensione nelle varie fasi del test.
Assumendo che i valori della tensione di cella alla fine di ciascuna fase di rilassamento siano pari ai valori dell'OCV, resta da determinare per tali punti il valore del SoC. Ciò è possibile a partire dal rapporto tra carica erogata/assorbita dalla cella (ottenuta per integrazione della corrente) e carica totale estratta della cella. Si ottengono così 20 punti per la carica e 20 punti per la scarica; tali punti, interpolati in matlab tramite la funzione "spline", forniscono la caratteristica SoC-OCV riportata in Figura 5.2.
Si può notare nel grafico la presenza del fenomeno di isteresi, anche se tale fenomeno non è molto significativo in quanto le curve SoC-OCV in carica e in

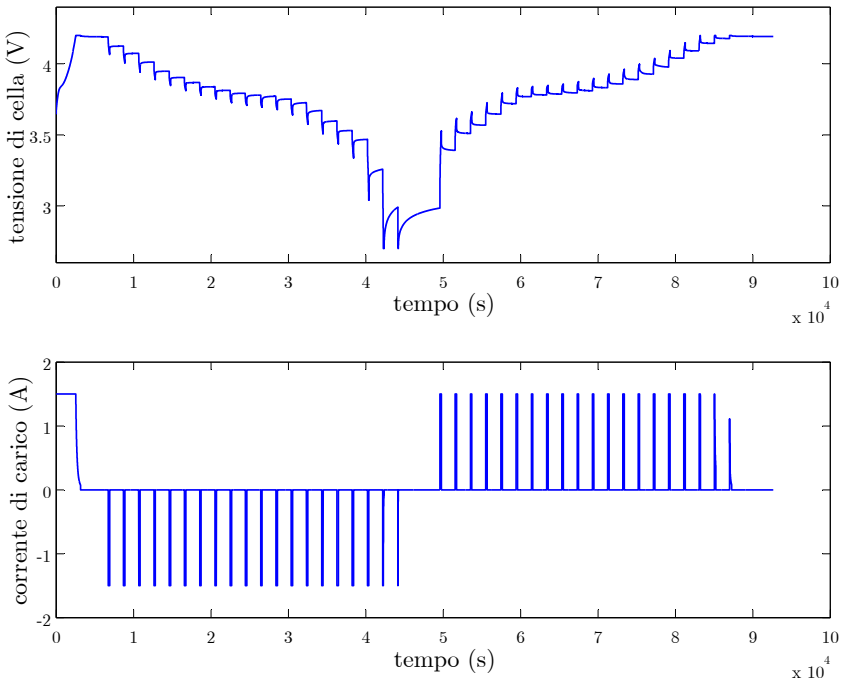

Figura 5.1: Andamento della tensione e della corrente durante il test

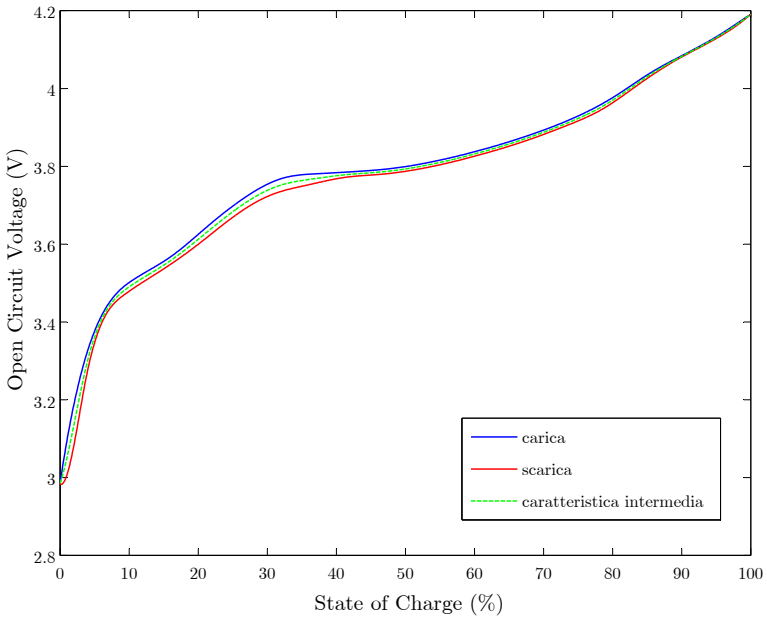

Figura 5.2: Grafico della caratteristica SoC-OCV

scarica differiscono al più di qualche mV. Dunque per una cella di questo tipo è possibile trascurare l'effetto dell'isteresi e utilizzare come curva SoC-OCV la media tra le curve di carica e scarica, ovvero la curva verde di Figura 5.2.

5.1.3 Misura offline dei parametri della cella

Dallo stesso test descritto nel paragrafo precedente è possibile estrarre una stima offline dei parametri della cella, utile in fase di implementazione del modulo di stima del SoC, il quale utilizza, in questa implementazione, un semplice modello circuitale a parametri costanti. L'algoritmo di stima completo, comprensivo della parte di identificazione dei parametri, non necessiterà di questi dati, in quanto i parametri verranno determinati direttamente a tempo di esecuzione del sistema.
Per estrarre i parametri è necessario analizzare l'andamento della tensione della batteria in condizioni di carica, scarica e rilassamento per diversi valori dello stato di carica, in quanto il valore dei parametri è funzione del SoC. In particolare si riescono a ricavare informazioni sui valori delle resistenze R_0 e R dalla risposta del sistema a gradini di corrente e il valore della costante di tempo $\tau = RC$ dall'andamento della tensione in fase di rilassamento, in quanto in tale condizione l'andamento della tensione è approssimabile ad un esponenziale. Il test per la misura della caratteristica SoC-OCV è dunque adatto all'estrazione dei parametri della cella. Osservando il circuito equiva-

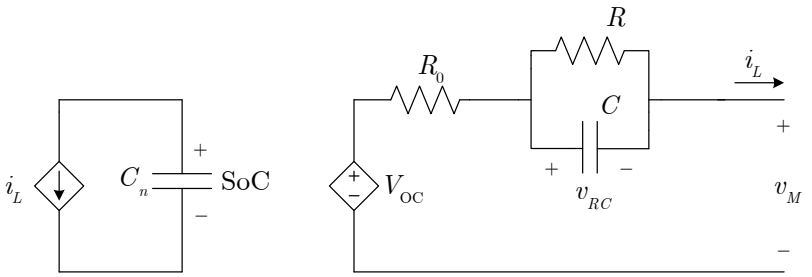

Figura 5.3: Circuito elettrico equivalente (modello ridotto)

lente ridotto, ovvero costituito da una sola squadra RC, descritto nel capitolo

3 (Figura 5.3), e ricordando che

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t} v_{RC}(t) = -\dfrac{v_{RC}(t)}{RC} + \dfrac{i_L(t)}{C} \\ v_M(t) = V_{\mathrm{OC}} - R_0 i_L(t) - v_{RC}(t) \end{cases}$$

si può determinare l'andamento della tensione di uscita del modello in risposta ad una corrente a gradini come quella di Figura 5.1 e confrontare il risultato con l'andamento reale (misurato) della tensione per estrarre i parametri, come descritto in [32].

In Figura 5.4 è mostrato un particolare dell'andamento della tensione di

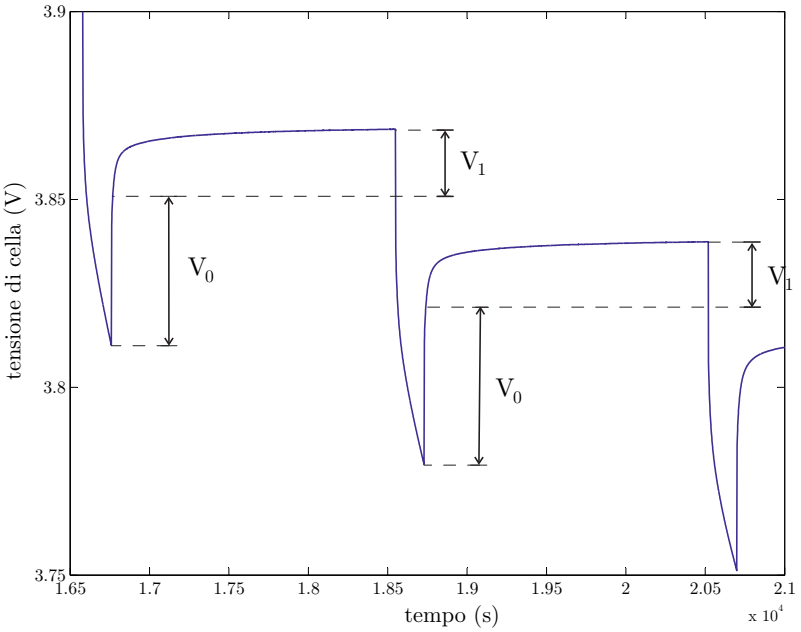

Figura 5.4: Andamento esponenziale della tensione di cella durante cicli di scarica e pausa

cella di Figura 5.1 nella prima parte (dove si alternano fasi di scarica a fasi di rilassamento). La tensione nelle due fasi ha un andamento simile ad un esponenziale, il quale è riprodotto nel modello dalla squadra RC; durante il passaggio da una fase di scarica ad una fase di rilassamento l'andamento della tensione è costituito da un tratto verticale, che tiene conto del cambiamento discontinuo della corrente (che passa da $I_0 = \mathrm{C}/20$ in scarica a 0), e da un

tratto simile ad un esponenziale che tiene conto dell'effetto di rilassamento. Fornendo in ingresso al modello circuitale un gradino di corrente si ottiene una risposta costituita da una componente a gradino pari a $V_0 = R_0 I_0$ ed una componente esponenziale pari a $V_1 (1 - e^{-t/\tau})$, dove V_1 è la differenza tra la V_{OC} per quel determinato valore di SoC e la tensione raggiunta dalla cella un istante dopo che la corrente è passata a 0.

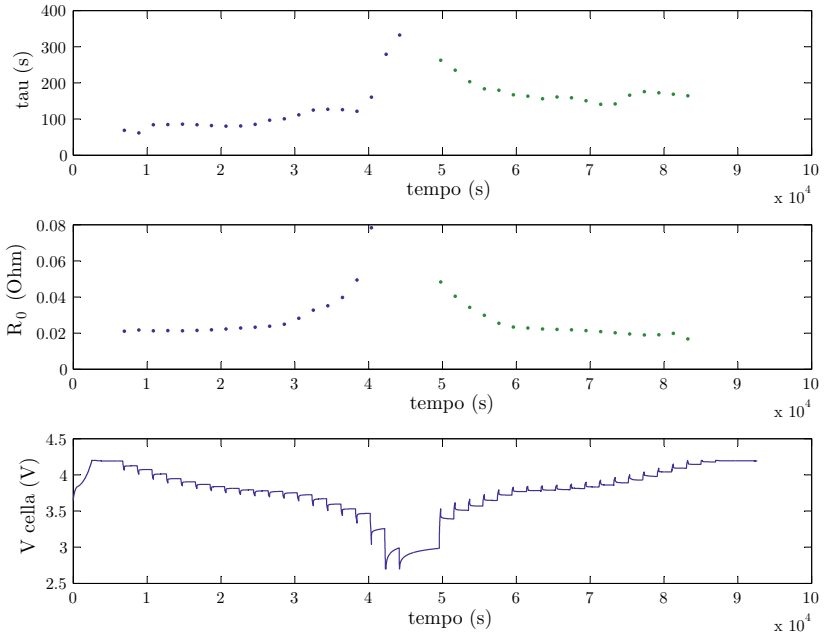

Figura 5.5: Valori di τ e R_0 calcolati

Risulta dunque evidente come estrarre i parametri. Per il calcolo di R_0 è sufficiente dividere il valore del salto di tensione V_0 misurato per I_0. Per ricavare il valore di V_1 e τ è stata utilizzata la funzione di fitting esponenziale di MATLAB (utilizzando un unico esponenziale). Una volta noti V_1 e τ si determinano R e C sapendo che $R = V_1/I_0$ (nell'ipotesi che alla fine dell'impulso di corrente siamo sia stata raggiunta una condizione stazionaria) e che $C = \tau/R$.
In Figura 5.5 sono riportati i valori di τ e di R_0 così calcolati, mentre in Figura 5.6 sono riportati i valori di R e C.

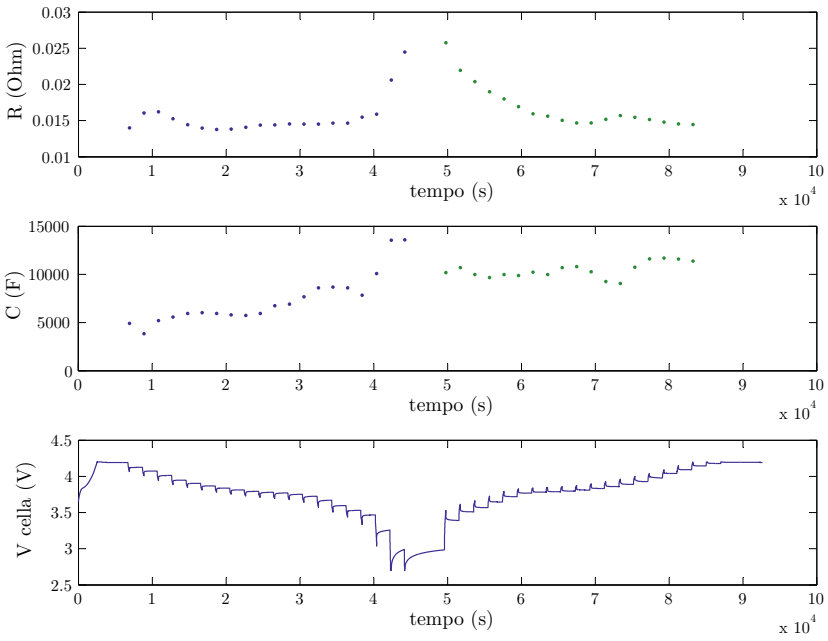

Figura 5.6: Valori di R e C calcolati

Per i test effettuati sulla cella che verranno descritti in seguito sono stati utilizzati valori di R_0, R e C costanti e pari al valor medio dei parametri calcolati, ovvero:

- $R_0 = 0.0263\,\Omega$
- $R = 0.0161\,\Omega$
- $C = 8814$ F

5.2 Set di verifica utilizzato

Viene qui presentata una breve descrizione del set di verifica impiegato nei test che verranno descritti in seguito.

5.2.1 Struttura generale

In Figura 5.7 è riportato uno schema sulla struttura complessiva del sistema realizzato. Come si può notare dalla figura, il modulo di stima del SoC è stato incluso all'interno dell'FPGA come modulo hardware memory-mapped che si interfaccia al processore tramite l'Avalon Interconnect Fabric. Per ottenere

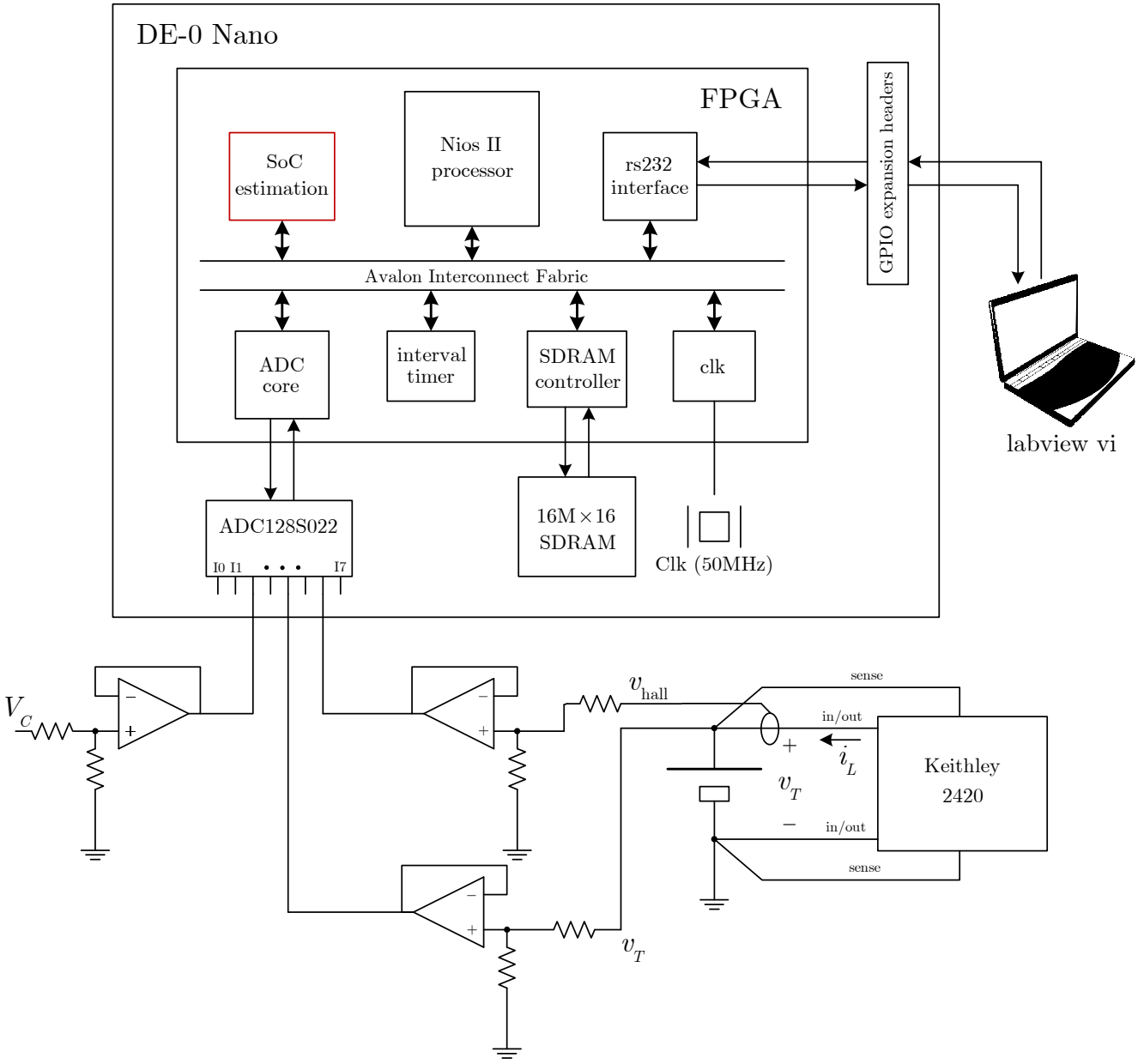

Figura 5.7: schema generale del sistema realizzato

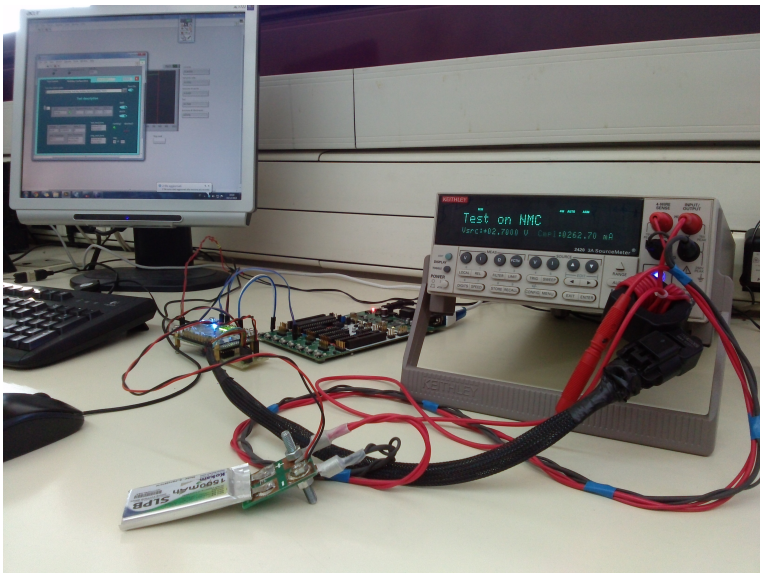

Figura 5.8: Test su cella in esecuzione

questo risultato il codice VHDL prodotto da DSP builder è stato importato in Quartus II ed è stato incluso in una entity superiore di interfacciamento con l'Avalon Interconnect Fabric, ovvero un'interfaccia Avalon-MM anch'essa descritta in VHDL. Successivamente tramite il tool Qsys di Quartus II è stato generato un modulo slave di stima del SoC ed incluso nel nios_system.
Assieme al modulo di stima del SoC nel nios_system sono istanziati altri moduli (IP Altera), tra cui i principali sono:

- un modulo di controllo della SDRAM presente sulla DE-0 che conterrà il software;
- un modulo di interfacciamento per la UART, ovvero per la comunicazione seriale con il PC;
- un modulo di controllo per l'ADC, necessario per acquisire i dati da elaborare;
- un modulo di *interval timer*, necessario a coordinare l'acquisizione di dati dall'ADC.

Tutti questi moduli sono gestiti dal software del processore.
Durante un test, il Keithley 2420 impone la corrente o la tensione di test

sulla cella e contemporaneamente ne misura i valori, i quali sono salvati sul PC in files di log. Tramite un sensore di Hall di corrente (DHAB s/25) si misura la corrente di cella. Tale sensore fornisce una tensione di uscita v_{hall} compresa tra 0 e 5 V proporzionale alla corrente, per cui essa, per poter essere acquisita dall'ADC (che ha dinamica di ingresso (0 - 3.3 V), dovrà essere ridotta tramite un partitore; inoltre si aggiunge anche un buffer per fornire all'ADC una sorgente di segnale a bassa impedenza, richiesta per il corretto funzionamento dell'ADC.
Allo stesso modo vengono acquisiti dall'ADC la tensione di cella v_T e la tensione di alimentazione V_C (5 V) del sensore di Hall. Quest'ultimo dato è necessario per poter determinare la corrente di cella a partire dal valore di v_{hall} in quanto il sensore è raziometrico, per cui la corrente sarà data dalla formula:

$$i_L = \left(v_{\text{hall}} - \frac{V_C}{2}\right) \frac{5}{G \cdot V_C} \tag{5.1}$$

dove G è la sensibilità.
Una volta acquisiti questi dati, essi sono inviati al modulo di stima del SoC, il quale li elabora e fornisce in uscita il valore del SoC stimato e il valore della tensione di uscita del modello. Questi dati, assieme ai dati acquisiti dall'ADC, vengono inviati tramite seriale al PC, dove è in esecuzione un programma Labview che provvede a mostrare i dati acquisiti e memorizzarli.

5.2.2 Descrizione del firmware

Di seguito è riportata una breve descrizione sul firmware impiegato.
Dopo aver inizializzato i moduli di stima del SoC, di interfaccia per l'UART e di controllo dell'ADC, il sistema attende di ricevere un segnale di inizializzazione da PC. Questo segnale, costituito da 6 stringhe, contiene i dati relativi alla cella necessari per l'inizializzazione del modulo di stima del SoC, che ancora non è stato abilitato. Questi dati sono i valori di R_0, R, C, T, SoC_{init} e Q_R, che vengono inviati come numeri in esadecimale in formato stringa.
Successivamente viene avviata la funzione di `alt_alarm` e viene abilitato il modulo di stima del SoC. La funzione `alt_alarm` dell' Hardware Abstraction Layer (HAL) mette il sistema in attesa di un certo tempo, corrispondente ad un certo numero di *clock ticks*, dopo il quale viene chiamata una certa

funzione di callback, definibile dal programmatore. Questo servizio offerto dall'HAL consente quindi di far eseguire funzioni a specifici istanti di tempo; per poter usufruire di questo servizio è necessario però che il sistema contenga una periferica timer in hardware: questa è costituita appunto dall'interval timer presente in Figura 5.7.
Il tempo stabilito per l'avvio della funzione di callback è 100ms: ogni 100ms viene chiamata una funzione la quale svolge le seguenti azioni:

1. acquisire i dati di corrente, tensione di cella e tensione di riferimento provenienti dall'ADC;
2. leggere i valori di SoC e di tensione di uscita del modello dal modulo di stima del SoC;
3. calcolare il valore di corrente secondo la relazione 5.1 e inviarlo al modulo di stima del SoC;
4. calcolare il valore della tensione di cella (dipendente dal rapporto di partizione e dal LSB dell'ADC) e inviarlo al modulo di stima del SoC;
5. inviare al PC tramite UART i dati di corrente, tensione di cella, tensione di uscita del modello, SoC stimato e tensione di riferimento, tutti in formato stringa.

Tutte queste operazioni vengono svolte ciclicamente.
Poiché la funzione di callback è chiamata a intervalli di tempo regolari e poiché in ciascuna chiamata il sistema effettua una acquisizione dei dati di corrente e tensione dall'ADC, di fatto il periodo di campionamento con cui opera il sistema di stima è dato dalla temporizzazione dell'`alt_alarm` e quindi pari a 100ms (comunque modificabile via software).

E' importante sottolineare che la presenza della componente software nel processo di stima del SoC non è indispensabile, ma in questo caso è stata utilizzata per semplicità realizzativa. Infatti, come già detto in precedenza, l'obiettivo di questo sistema è quello di svincolare la stima del SoC dalla componente software; poiché il software qui utilizzato si occupa semplicemente di temporizzare l'acquisizione dei dati e di inviarli al blocco di stima del SoC, queste funzioni possono essere svolte semplicemente da un DMA oppure da logica custom opportunamente progettata.

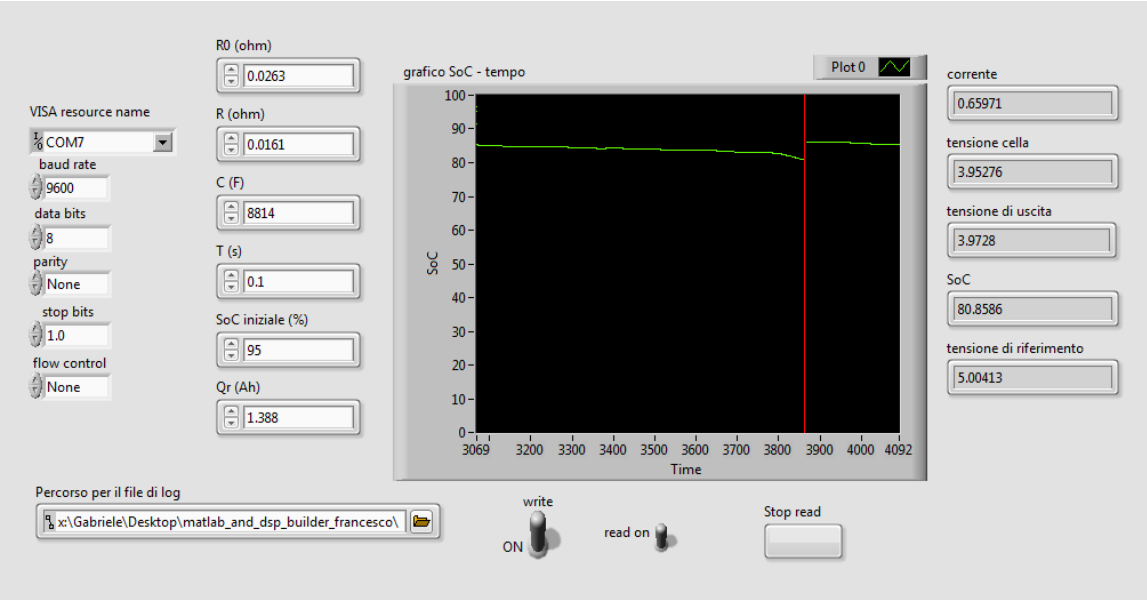

Figura 5.9: Front panel del VI in Labview

5.2.3 Interfaccia in Labview

Come già anticipato nei precedenti paragrafi, il sistema comunica con un virtual instrument (VI) in Labview; il front panel del VI è mostrato in Figura 5.9. Questa interfaccia non effettua calcoli o elaborazioni sui dati che riceve dal sistema, ma provvede semplicemente a mostrarli su schermo e a memorizzarli.

Come si nota dalla figura, il front panel è costituito da una parte di impostazione della comunicazione seriale, che consente di specificare la porta di comunicazione (COM) e le altre impostazioni di comunicazione quali baud rate, numero dei bit per ogni dato, presenza del bit di parità e numero di stop bits, e da una parte di inizializzazione del modulo di stima del SoC, che consente di assegnare i valori di R_0, R, C, T, $\mathrm{SoC}_{\mathrm{init}}$ e Q_R da inviare al modulo. La parte destra del front panel, invece, mostra i valori dei dati ricevuti dal sistema; per quanto detto in precedenza questi dati vengono inviati con una frequenza di 10 Hz. Inoltre, il valore del SoC calcolato dal modulo è riportato in un grafico in funzione del tempo che si aggiorna durante l'esecuzione del programma.

Tutti i dati ricevuti sono memorizzati in un file di log, grazie al quale sarà

possibile confrontarli con i dati acquisiti dal Keithley 2420.

5.3 Tests di verifica

In questo paragrafo sono descritti i tests effettuati e sono riportati i risultati ottenuti, confrontati sia con gli esiti delle simulazioni in Simulink, sia con i risultati attesi elaboarati a partire dalle misure del Keithley 2420.

5.3.1 Urban Dynamometer Driving Schedule (UDDS)

Per la scelta del tipo di test e delle condizioni di funzionamento della cella che risultino adatte a consentire una buona valutazione dell'algoritmo implementato è necessario stabilire un benchmark rappresentativo delle condizioni operative della cella, in particolare quando essa alimenta un veicolo elettrico.
A questo scopo, una possibile scelta è costituita dai cicli di guida standard

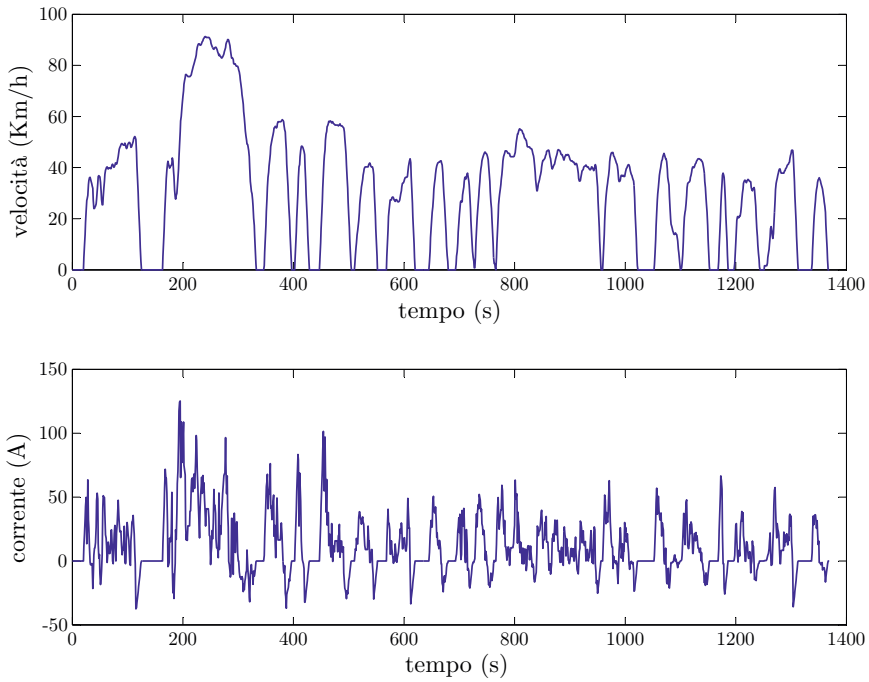

Figura 5.10: Profilo di velocità e corrente dell'UDDS

utilizzati per testare le autovetture valutandone i livelli di emissione del motore a combustione interna e il risparmio di carburante. Uno di questi è lo Urban Dynamometer Driving Schedule (UDDS) [33], definito dalla U.S. Environmental Protection Agency. Un ciclo di UDDS simula un percorso urbano di 12.07 Km della durata di circa 22 minuti con fermate frequenti; la massima velocità è di 91.25 Km/h e la velocità media è di 31.5 Km/h. In Figura 5.10 sono riportati il profilo di velocità dell'UDDS e il corrispondente profilo di corrente per una batteria da 66.2 Ah, derivato secondo il procedimento descritto in [11].

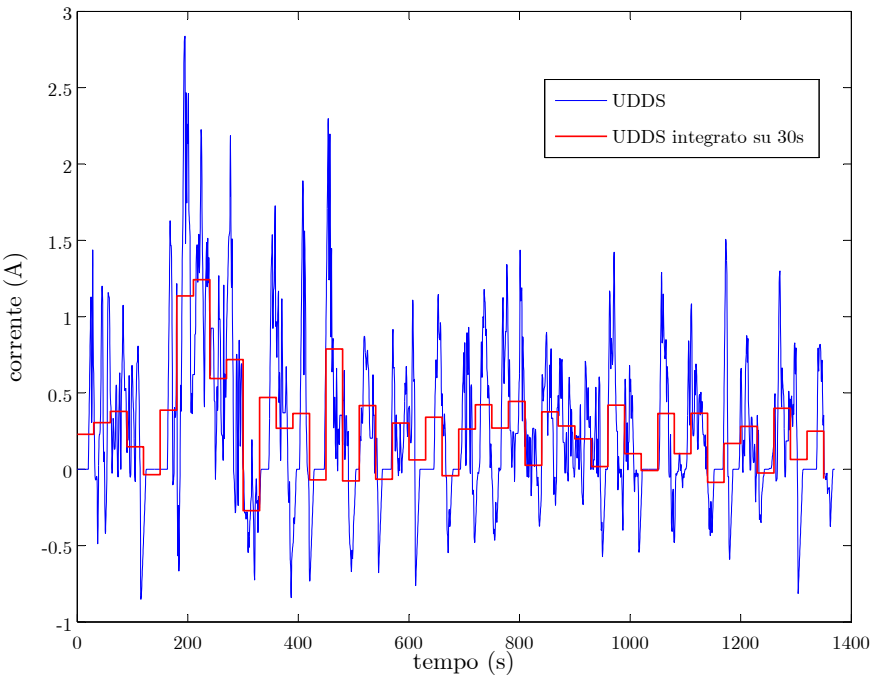

Figura 5.11: Calcolo della corrente UDDS di test

Nei test effettuati, dunque, sono stati imposti alla cella più cicli di UDDS, avendo opportunamente scalato il valore della corrente di test in proporzione alla capacità complessiva della cella (pari a 1.5 Ah). Avendo a disposizione come generatore/carico della batteria il Keithley 2420, si è deciso di ricostruire il ciclo di corrente UDDS come una funzione a scalini: il valore di

ogni scalino corrisponde al valore di corrente imposto sulla cella dallo strumento. Tuttavia non è stato possibile imporre un UDDS identico a quello di Figura 5.10 in quanto sia a causa del tipo di controllo (tramite un VI di Labview) sia a causa delle caratteristiche dello strumento si ha che durante il cambiamento della corrente da un valore ad un altro la corrente di uscita risulta nulla per un intervallo di tempo circa pari a 0.6 - 0.7 secondi.
La soluzione è stata dunque quella di generare una funzione a scalini calcolata tramite integrazioni della corrente UDDS su finestre temporali di 30 s, come mostrato in Figura 5.11. In questo modo il tempo in cui la corrente risulta nulla è trascurabile rispetto alla durata dell'intervallo temporale durante il quale la corrente è mantenuta costante.

5.3.2 Test con cicli UDDS ripetuti

Un primo test utile per valutare la qualità dell'algoritmo è quello di imporre alla cella un certo numero di cicli UDDS, facendo sì che essa si scarichi in maniera significativa. A tale scopo si effettua un test in cui si carica la cella fino al valore del 100% di SoC, in quanto ciò consente di portarsi in uno stato noto; in seguito si impongono alla cella 12 cicli di UDDS consecutivi in modo da scaricarla quasi completamente (fino ad un valore di circa il 10% del SoC) e infine si completa la scarica a corrente costante fino a raggiungere lo 0% del SoC. Dopo una breve pausa si ricarica completamente la cella. L'andamento della corrente di test e della tensione di cella sono riportati in Figura 5.12.
La scelta di caricare completamente la cella nella prima fase e successivamente di scaricarla completemente è dovuta al fatto che per poter effettuare un calcolo accurato del valore reale di SoC, da usare come termine di paragone, è necessario conoscere due dati con precisione:

1. il valore iniziale dello stato di carica;
2. la carica massima effettivamente estraibile dalla cella (Q_R).

Per conoscere il valore iniziale dello stato di carica è necessario portarsi in una condizione nota, da cui la scelta di portarsi nella condizione iniziale di "cella completamente carica". Inoltre, la carica massima estraibile, poiché varia a seconda delle condizioni operative della cella e dell'invecchiamento,

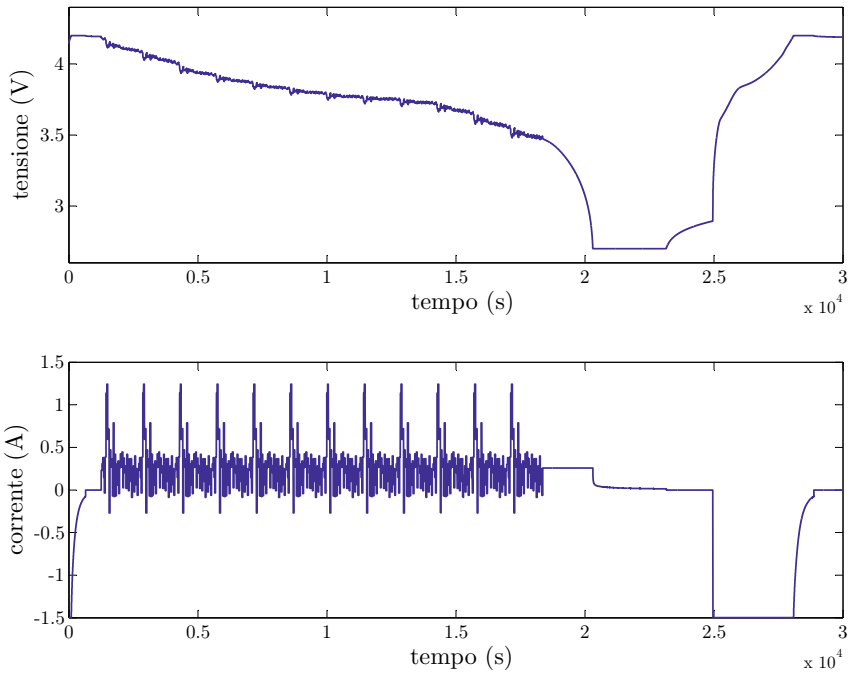

Figura 5.12: Tensione e corrente di test

risulterà essere diversa dalla carica nominale Q_n, per cui il modo più semplice di determinarla consiste nell'effettuare una misura diretta, integrando la corrente estratta in una scarica completa della cella.
In questo modo è possibile ottenere l'andamento del SoC di riferimento in funzione del tempo semplicemente integrando la corrente misurata dal Keithley 2420:

$$\mathrm{SoC}(t) = \mathrm{SoC}_{\mathrm{init}} + \frac{1}{Q_R} \int_{t_0}^{t} i_L(\tau)\, \mathrm{d}\tau$$

ovvero applicando la tecnica del Coulomb Counting. Non si hanno in questo caso i problemi che solitamente caratterizzano questa tecnica in quanto lo stato iniziale di SoC è noto, quindi non c'è errore sul $\mathrm{SoC}_{\mathrm{init}}$, e la misura di corrente è accurata perché eseguita da uno strumento di misura affidabile.

Nelle figure 5.13 e 5.15 sono riportati i risultati ottenuti dal test. In Figura 5.13 è riportato l'andamento della tensione di cella durante la fase di scarica con cicli di UDDS e la tensione di uscita del modello (in verde); si

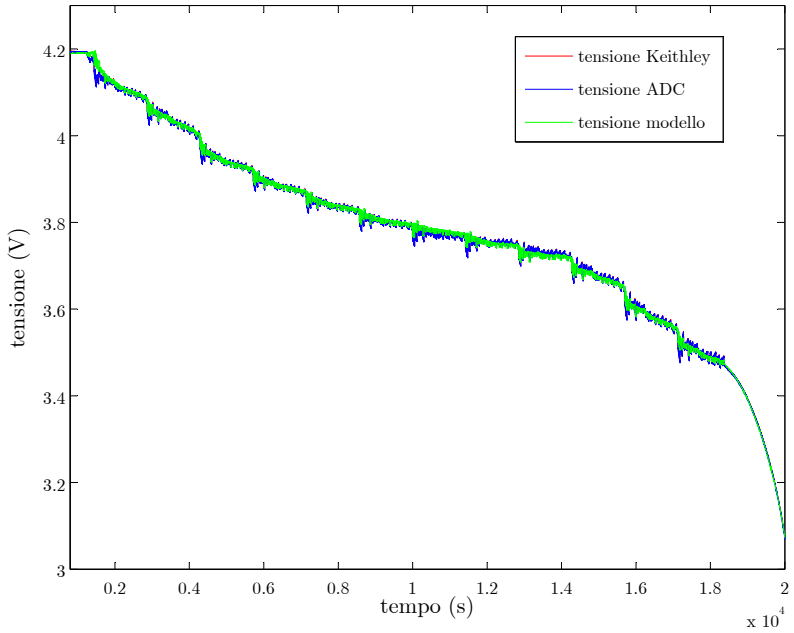

Figura 5.13: confronto tra tensione di uscita del modello e tensioni acquisite

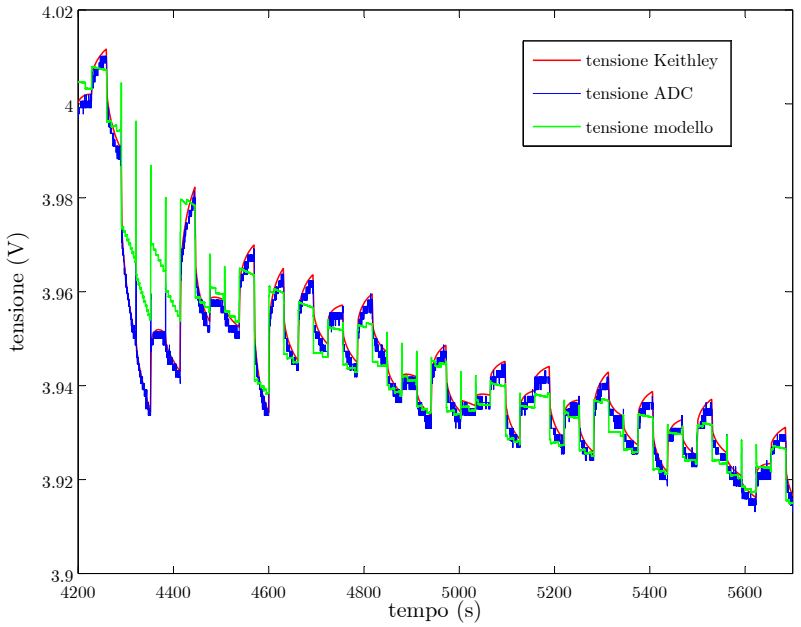

Figura 5.14: Confronto tra tensione di uscita del modello e tensioni acquisite (ingrandimento)

nota che essa segue bene la tensione reale di cella. La presenza di picchi verso l'alto, presenti sia nella tensione di uscita del modello sia nella tensione acquisita dall'ADC ma non in quella acquisita dal Keithley, sono dovuti all'evento descritto in precedenza di annullamento momentaneo della corrente che si verifica ogni volta che viene imposto un nuovo valore di corrente. Il Keithley non può rilevare la presenza di tali picchi sulla tensione di cella in quanto esso può effettuare il campionamento ad una frequenza massima di 1 Hz.

In Figura 5.15 è riportato il SoC stimato dal sistema e il SoC di riferimento.

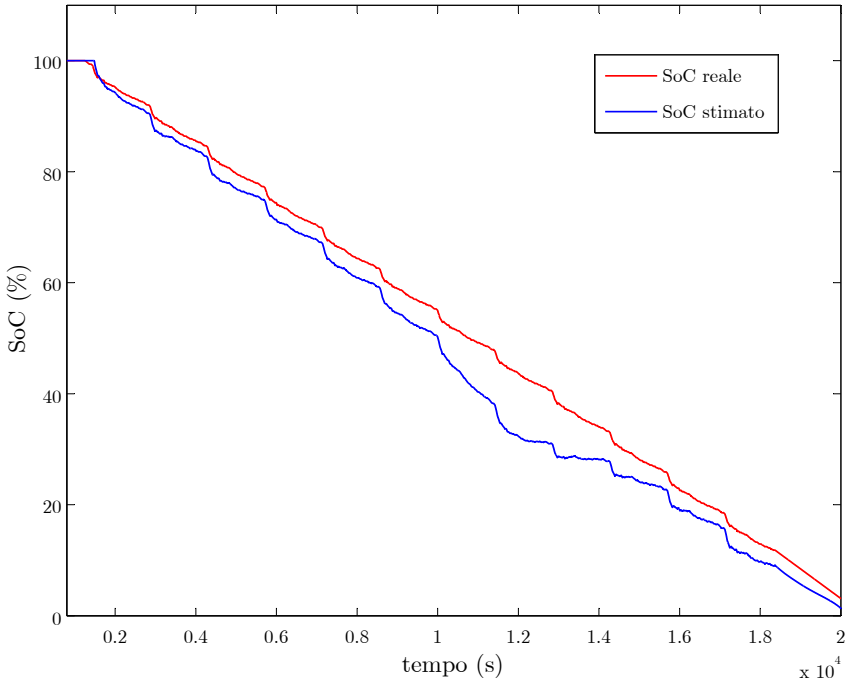

Figura 5.15: Confronto fra SoC stimato e SoC reale

Si nota che il SoC stimato segue abbastanza bene il SoC reale ad eccezione della zona centrale, ovvero per valori di SoC attorno al 40%-50%. In questa zona il SoC stimato differisce da quello reale di una quantità abbastanza significativa (attorno al 10% di SoC). In Figura 5.16 è riportato l'andamento dell'errore assoluto sulla stima del SoC, ovvero $\mathrm{SoC_{err}} = \mathrm{SoC_{reale}} - \mathrm{SoC_{stimato}}$ rispetto al $\mathrm{SoC_{reale}}$. Risulta adesso evidente la presenza di un picco dell'errore

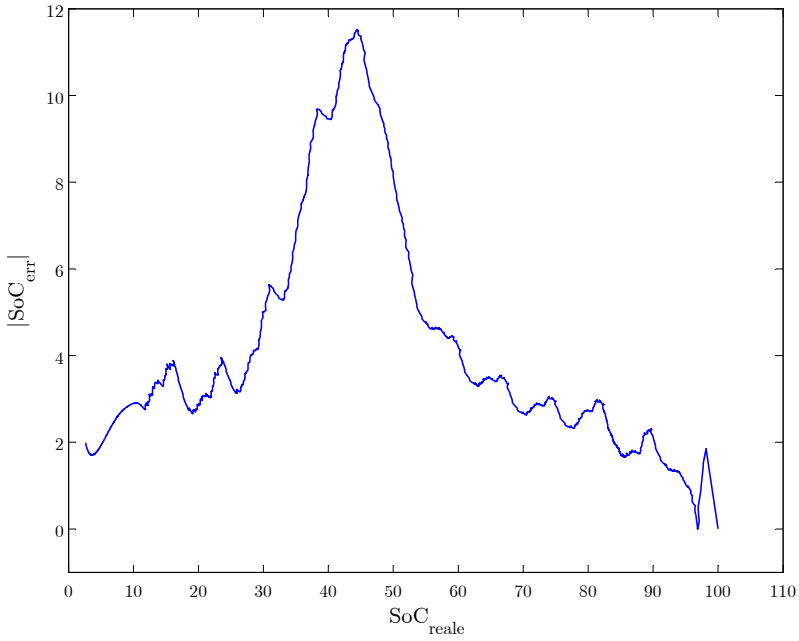

Figura 5.16: Errore assoluto sulla stima del SoC

in una zona intorno al 40%-50% del SoC reale.
Per valutare quantitativamente l'entità di questo errore è utile calcolare le seguenti quantità: il valore RMS dell'errore (root mean square error, RMSE) e l'errore massimo E_{max}, definiti come:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \text{SoC}_{\text{err}}^2(k)}$$

$$E_{max} = \max_k |\text{SoC}_{\text{err}}(k)|$$

Nel caso del test in questione si ha:

- RMSE = 4.95%
- E_{max} = 11.52%

L'errore sulla stima del SoC e in particolare il suo andamento per valori di SoC attorno al 40%-50% è dovuto sostanzialmente a due cause:

1. il modello della cella non rappresenta accuratamente il comportamento reale della cella;

2. il tempo di risposta del sistema aumenta a causa del fatto che la caratteristica non lineare SoC-OCV per valori di SoC attorno al 40% ha una pendenza molto ridotta (è quasi piatta).

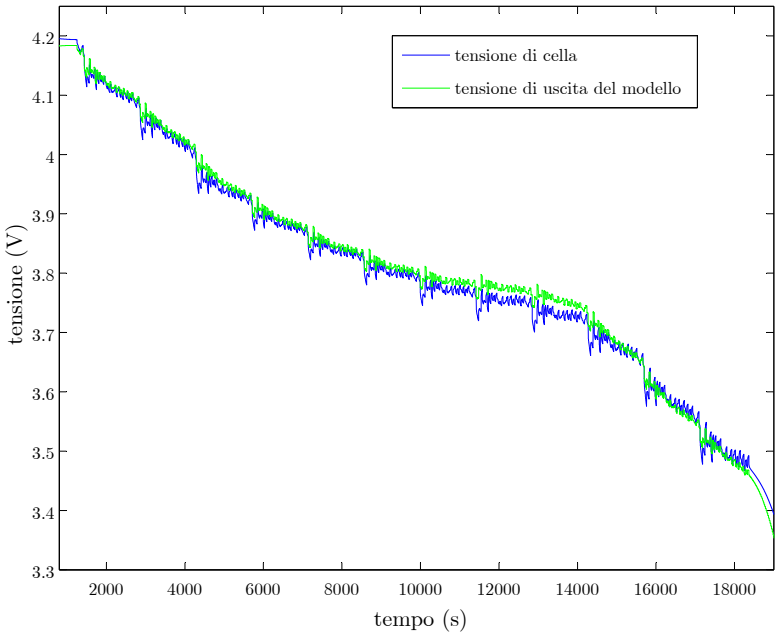

Figura 5.17: Confronto fra tensione di cella misurata e tensione di uscita del modello ad anello aperto

Per quanto riguarda il punto 1, esso è dovuto a diversi fattori, ma principalmente al fatto che il modello utilizzato è a parametri costanti. Se esso fosse a parametri variabili, aggiornati a tempo di esecuzione tramite il processo di identificazione on-line, si avrebbe un notevole miglioramento dovuto al fatto che i parametri del modello si adeguerebbero alle variazioni dello stato della cella.
Utilizzando invece il modello a parametri costanti si ottiene il risultato di Figura 5.17, dove è mostrato il confronto tra la tensione di cella misurata (in blu) e la tensione di uscita del modello ad anello aperto (in verde). Si nota che la tensione del modello si discosta da quella reale in particolare nella zona centrale corrispondente alla zona "quasi piatta" della caratteristica

SoC-OCV.

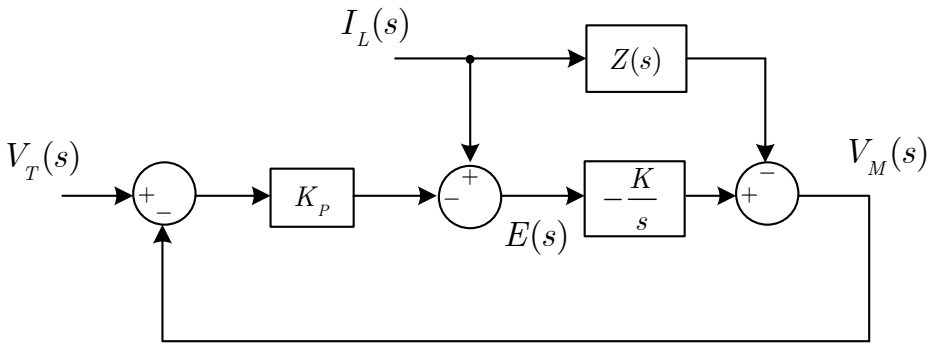

Figura 5.18: Schema a blocchi semplificato

Per quanto riguarda invece il tempo di risposta del sistema, esso può essere determinato a partire dalla funzione di trasferimento del sistema. Considerando lo schema a blocchi linearizzato di Figura 3.4 descritto nel capitolo 3, esso può essere riadattato nella versione semplificata di Figura 5.18, nella quale:

$$K = \frac{1}{Q_R} \cdot \left.\frac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{dSoC}}\right|_{\text{punto di lavoro}} = \frac{\alpha_1}{Q_R}$$
$$Z(s) = R_0 + \frac{R}{1+RCs} = \frac{(R_0+R)+R_0RCs}{1+RCs}$$

dunque si ha che:

$$\begin{cases} V_M(s) = -Z(s)I_L(s) - \dfrac{K}{s}\,E(s) \\ E(s) = I_L(s) - K_P\,\left(V_T(s) - V_M(s)\right) \end{cases}$$

$$
\begin{aligned}
V_M(s) &= -Z(s)I_L(s) - \frac{K}{s}\left[I_L(s) - K_P\left(V_T(s) - V_M(s)\right)\right] \\
V_M(s)\left(1 + \frac{K_P K}{s}\right) &= -\left(\frac{K}{s} + Z(s)\right) I_L(s) - \frac{K_P K}{s} V_T(s) \\
V_M(s) &= \frac{\frac{K_P K}{s}}{1 + \frac{K_P K}{s}} V_T(s) - \frac{\frac{K}{s} + Z(s)}{1 + \frac{K_P K}{s}} I_L(s)
\end{aligned}
$$

da cui:

$$
V_M(s) = \frac{1}{1 + \frac{s}{K_P K}} V_T(s) - \frac{1}{K_P} \frac{1 + s\frac{Z(s)}{K}}{1 + \frac{s}{K_P K}} I_L(s) \tag{5.2}
$$

Il SoC è invece dato da:

$$
\begin{aligned}
\mathrm{SoC}(s) &= -\frac{1}{Q_R}\frac{E(s)}{s} = -\frac{1}{Q_R s}\left[I_L(s) - K_P(V_T(s) - V_M(s))\right] \\
&= -\frac{1}{Q_R s}\left[I_L(s) - K_P V_T(s) + \frac{K_P^2 K}{s + K_P K} V_T(s) - K_P \frac{1 + sZ(s)/K}{1 + s/(K_P K)}\right] \\
&= \frac{1}{Q_R}\frac{K_P}{s + K_P K} V_T(s) + \frac{1}{Q_R}\frac{K_P Z(s) - 1}{s + K_P K} I_L(s)
\end{aligned}
$$

La costante di tempo del sistema dunque risulta:

$$
\tau_s = \frac{1}{K_P K} = \frac{Q_R\left(R_0 + R\right)}{\alpha_1} \tag{5.3}
$$

Se la pendenza della caratteristica SoC-OCV diminuisce (ovvero α_1 diminuisce) la costante di tempo del sistema aumenta. Se la costante di tempo aumenta in maniera considerevole, il sistema diventa lento e la tensione di uscita v_M non è più in grado di seguire la tensione v_T. In realtà il sistema funzionerebbe correttamente su tempi più lunghi, ma in tempi lunghi il SoC varia sensibilmente nelle condizioni di test imposte, per cui varia anche il punto di lavoro attorno cui si è linearizzato il sistema. Tutto ciò contribuisce a generare un errore sulla stima del SoC, in particolare nella zona "quasi piatta" della caratteristica SoC-OCV. Questo spiega dunque la presenza del picco dell'errore nel grafico di Figura 5.16. Si può dunque concludere che se nella zona in cui la curva SoC-OCV è piatta il sistema (cioè la variabile di stato SoC) non è osservabile.

Oltre al risultato in sé della stima resta da valutare un altro risultato fondamentale, ovvero la coerenza del funzionamento dal sistema mappato su FPGA rispetto alle simulazioni del sistema stesso svolte in Simulink. Ciò è essenziale come conferma della validità dello strumento di sintesi e del flusso di progetto adottato.

Nel caso del test in questione i risultati sono soddisfacenti: in Figura 5.19 (e

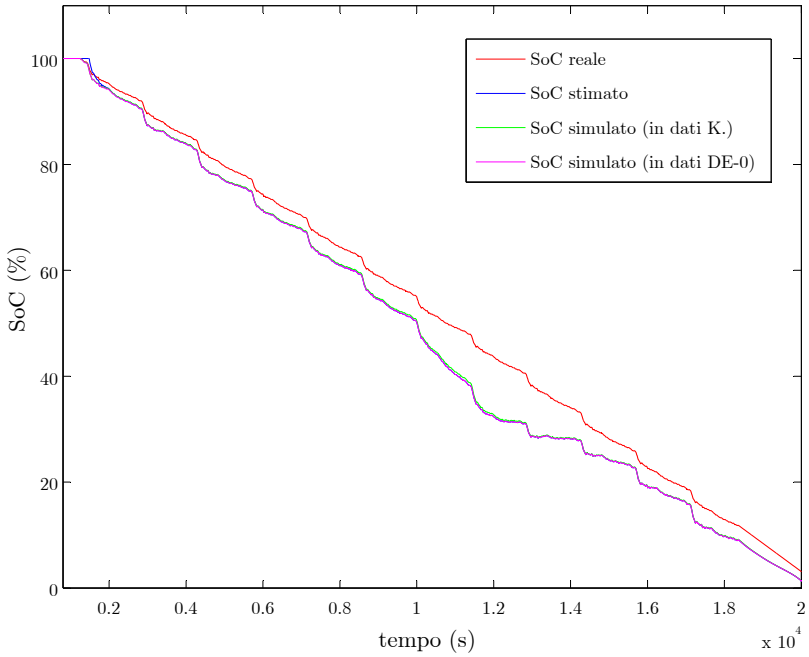

Figura 5.19: Confronto fra risultato sperimentale e simulazione

in Figura 5.20 che ne mostra un ingrandimento), sono riportati in uno stesso grafico i risultati del test eseguito e delle simulazioni svolte in Simulink. Per quanto riguarda le simulazioni, ne sono state eseguite due: una prima simulazione che riceve in ingresso i dati di tensione e corrente acquisiti dal Keithley 2420, ed una seconda simulazione che riceve in ingresso i dati acquisiti dall'ADC della DE-0.

Si nota che l'esito della simulazione risulta del tutto coerente con il risultato del test: le tre curve (le due di simulazione e quella di test) sono praticamente coincidenti, tanto che nel grafico di Figura 5.19 non si riescono quasi a distin-

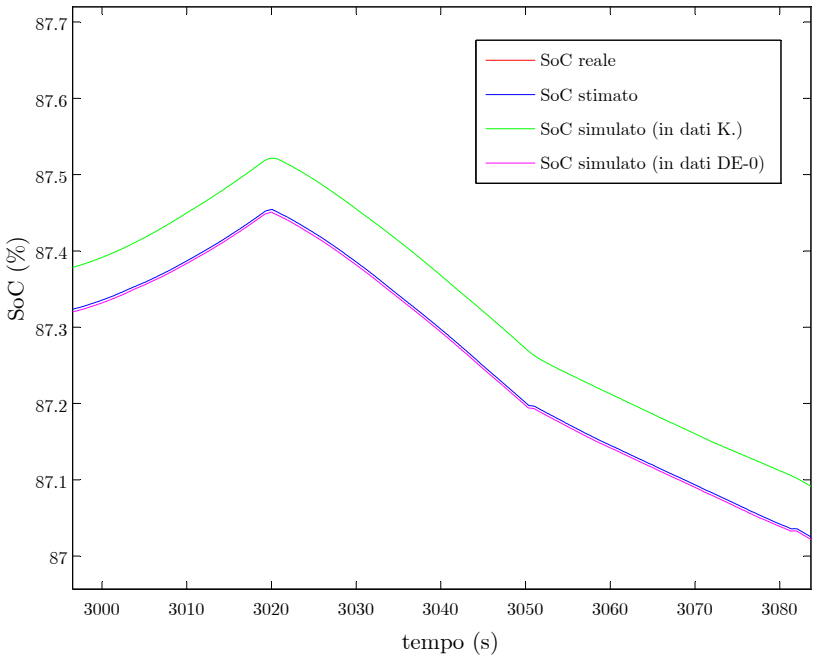

Figura 5.20: Confronto fra risultato sperimentale e simulazione (ingrandimento)

guere. Per questo motivo in Figura 5.20 è mostrato un ingrandimento, dal quale si nota che la curva relativa alla simulazione avente in ingresso i dati dell'ADC e quella relativa al risultato del test (la quale quindi è il risultato di un calcolo sugli stessi dati) differiscono di meno dello 0.01% del SoC.
Il risultato è dunque pienamente soddisfacente e costituisce dunque una conferma sulla validità del flusso di progetto adottato in questo lavoro.

5.3.3 Test con cicli UDDS ripetuti in presenza di offset

Questo secondo test è volto a valutare il comportamento del sistema in presenza di un offset sulla misura della corrente. Nel capitolo 3 è stata svolta una analisi teorica sul comportamento del sistema in presenza di offset, da cui risultava che per un valore di K_P pari a $1/(R_0 + R)$ si ottiene a regime (ovvero per un tempo $t \to \infty$) un annullamento dell'errore sulla stima del SoC dovuto alla presenza di un offset sulla misura della corrente.
Questo test è identico al precedente, l'unica differenza è costituita dalla presenza di un offset fittizio generato via software tramite l'aggiunta di una co-

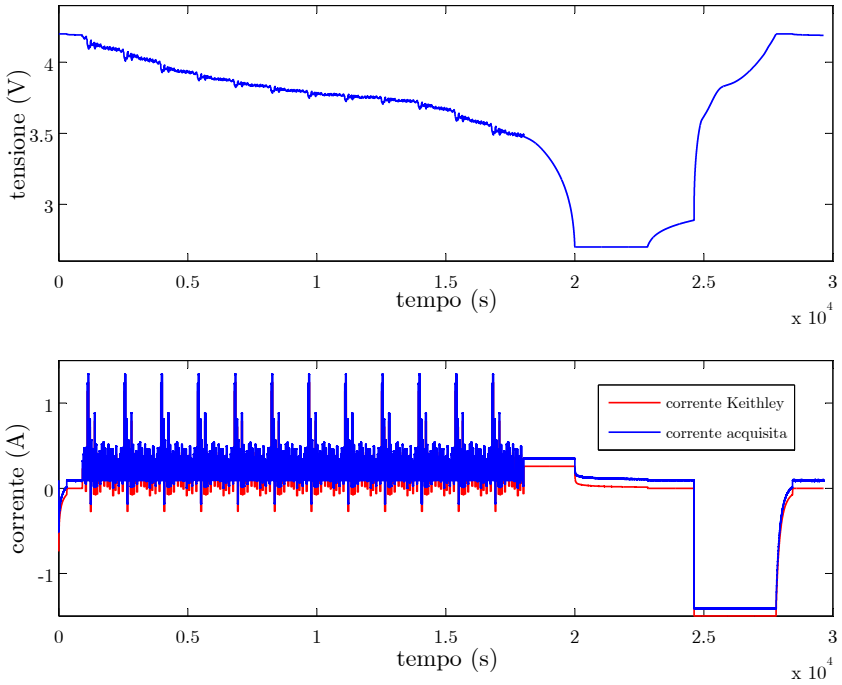

Figura 5.21: Tensione e corrente di test

stante ai dati di corrente acquisiti dall'ADC, come si nota dalla Figura 5.21. L'entità dell'offset è, in questo caso, pari a circa 90 mA, un valore piuttosto elevato se si pensa che il valore massimo di corrente di test è di 1.24 A.

In Figura 5.22 è riportato il SoC stimato durante il test assieme al SoC di riferimento (calcolato nello stesso modo del test precedente). Anche in questo caso si nota che si ha un discostamento del SoC stimato rispetto a quello di riferimento che risulta più significativo nella zona attorno al 40%-50% di SoC, esattamente come nel caso precedente. Inoltre questa differenza risulta più marcata rispetto al test precedente, ma non molto superiore e comunque non così significativa da compromettere la stima del SoC. Inoltre nello stesso grafico è riportato il SoC risultante dall'utilizzo del solo metodo del Coulomb-Counting, così da avere come termine di paragone uno dei metodi attualmente più utilizzati per la stima del SoC. Si nota che l'offset sulla corrente (in questo caso di valore elevato per rappresentare condizioni di funzionamento particolarmente avverse) nel metodo del Coulomb-Counting

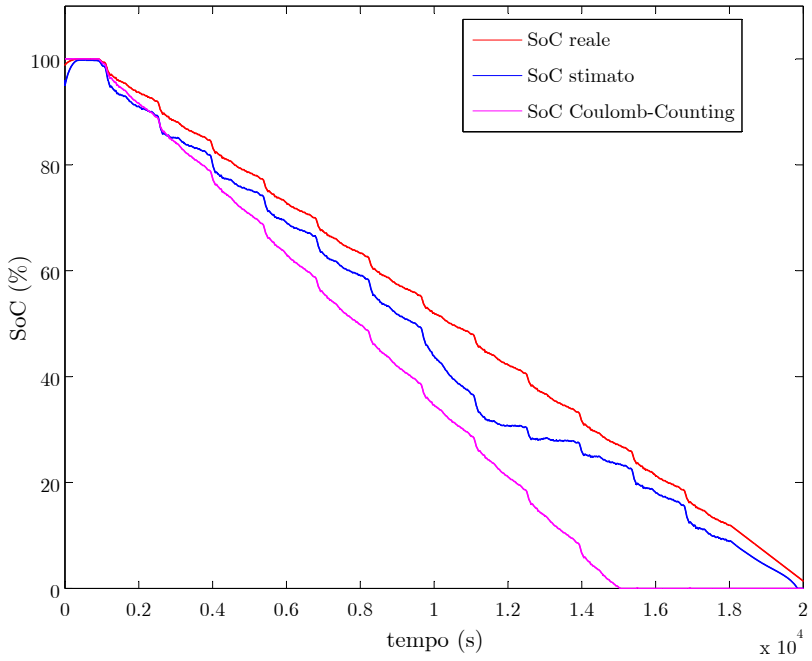

Figura 5.22: Confronto fra SoC stimato e SoC reale in presenza di offset sulla corrente

comporta un discostamento significativo e progressivo dal valore reale del SoC che non viene recuperato nel tempo.

In Figura 5.23 si può osservare l'andamento dell'errore assoluto sulla stima del SoC in funzione del SoC reale. Da un confronto con l'errore ottenuto nel precedente test, si nota che l'errore sulla stima del SoC risulta essere superiore, com'era ragionevole aspettarsi; ciò è infatti confermato dal valore del RMSE e dell'errore massimo

- RMSE $= 5.63\%$
- $E_{max} = 12.76\%$

i quali risultano essere entrambi superiori a quelli ottenuti in precedenza.

Questo comportamento è dovuto al fatto che il sistema, sebbene sia in grado di annullare l'errore sulla stima del SoC dovuto alla presenza di un offset sulla corrente per $K_P = K_{P_{\text{opt}}}$, annulla l'errore soltanto al tempo $t \to \infty$. Ciò significa che il sistema si comporta come un sistema del primo ordine

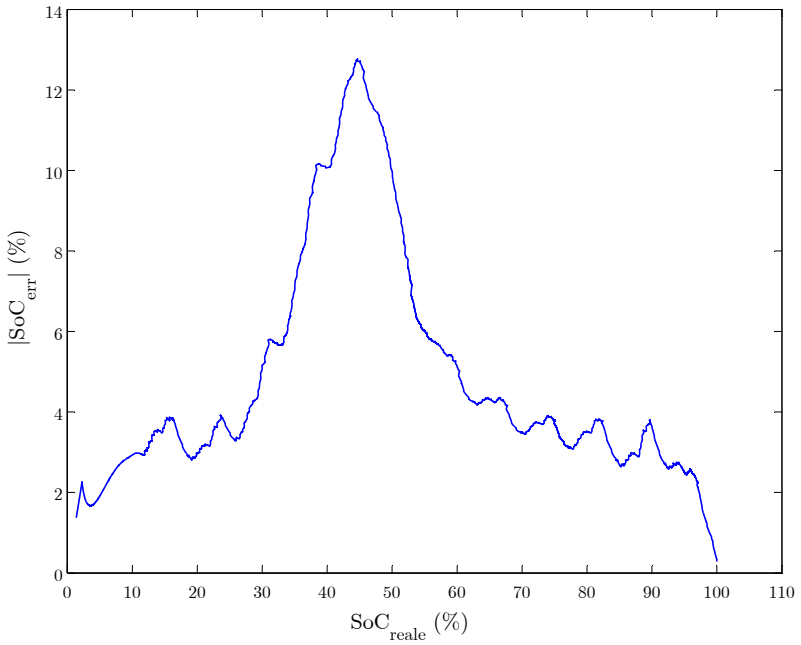

Figura 5.23: Errore assoluto sulla stima del SoC

con costante di tempo pari a

$$\tau_s = \frac{Q_R\,(R_0 + R)}{\alpha_1}$$

calcolata nel precedente paragrafo. Dunque il tempo che impiega il sistema a correggere l'errore sul SoC dovuto all'offset sulla corrente, che si somma all'errore già presente discusso in precedenza, è inversamente proporzionale ad α_1, ovvero alla pendenza della caratteristica SoC-OCV nel punto di lavoro. Risulta quindi evidente che quando la cella si trova in uno stato a cui corrisponde una pendenza bassa della caratteristica SoC-OCV l'errore risultante dovuto alla presenza dell'offset è maggiore che non nelle zone in cui la pendenza è minore.

Questo fatto risulta particolarmente evidente se si osserva la risposta del sistema per diversi valori di offset e la si confronta con la caratteristica SoC-OCV, riportata per semplicità in Figura 5.24. Purtroppo effettuare molti test con valori diversi di corrente di offset risulta molto dispendioso in termini di

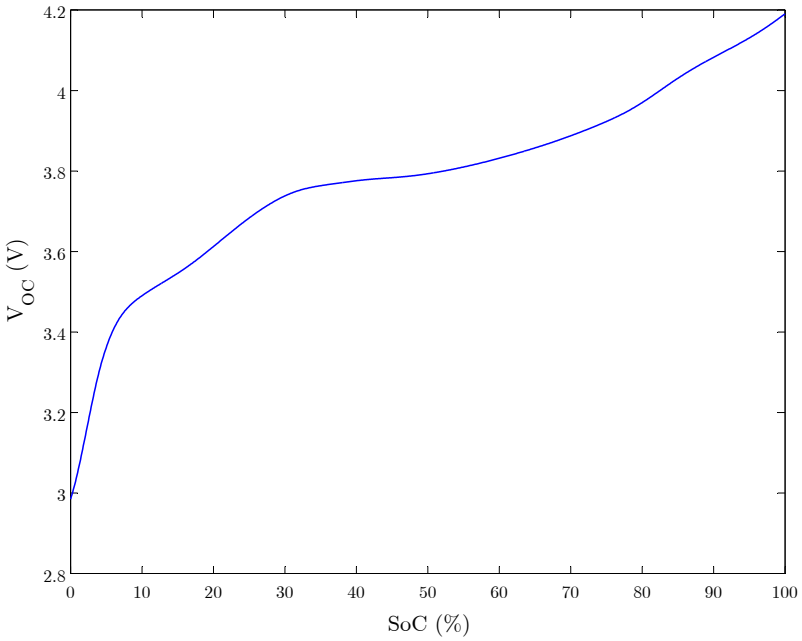

Figura 5.24: Caratteristica SoC-OCV SoC

tempo in quanto ciascun test ha una durata di circa 8 ore. Tuttavia, poiché nel precedente paragrafo è stato mostrato come i risultati delle simulazioni siano praticamente coincidenti con i risultati sperimentali, si è deciso di svolgere i test con diversi valori di offset soltanto a livello simulativo.

I risultati delle simulazioni sono riportati in Figura 5.25. Si nota che le curve si discostano tra loro e rispetto alla curva di riferimento in particolare nella zona attorno al 40% del SoC reale; tale zona corrisponde alla zona a pendenza minore della caratteristica SoC-OCV, come si osserva dalla Figura 5.24. Nelle altre zone, dove la pendenza della caratteristica è più elevata, le curve sono più ravvicinate; in particolare questo si nota nella zona finale, ovvero per valori di SoC reale inferiori al 20%, a cui infatti corrisponde una maggiore pendenza della caratteristica.

Questo risultato dunque costituisce una conferma del fatto che l'errore sulla stima del SoC dovuto alla presenza di offset è effettivamente causato da un rallentamento del sistema.

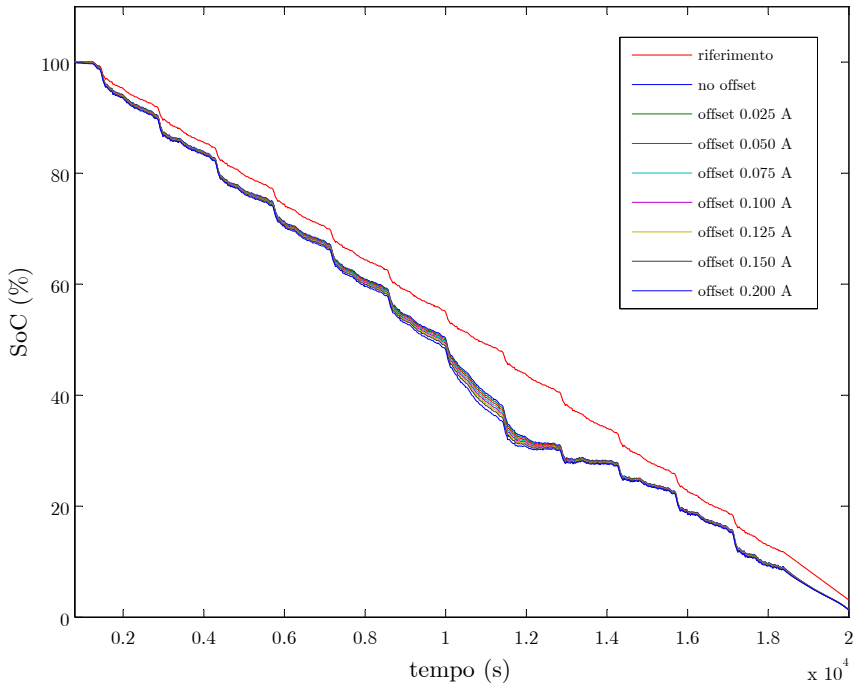

Figura 5.25: Risultato delle simulazioni svolte per diversi valori di offset di corrente

5.3.4 Test con cicli UDDS separati da pause

In questo test si è scelto di sottoporre la cella a cicli UDDS ripetuti ma separati da pause della durata di un'ora. In tali pause la corrente imposta alla cella è nulla, quindi la cella può rilassarsi e la sua tensione può tendere al valore dell'OCV. Questo test risulta utile in quanto consente sia di verificare se effettivamente l'errore sulla stima del SoC dovuto a un rallentamento del sistema viene compensato quando il sistema resta a riposo per un certo periodo di tempo, sia di verificare se effettivamente avviene una variazione del tempo di risposta del sistema al variare del SoC, come ricavato per via algebrica in precedenza.

In Figura 5.26 sono riportate la tensione e la corrente di test. Il test è costituito da 12 cicli UDDS separati da pause della durata di un'ora a corrente nulla; successivamente la cella viene scaricata completamente a C per poter determinare la carica massima da essa estraibile e infine, dopo una breve pausa, viene ricaricata completamente.

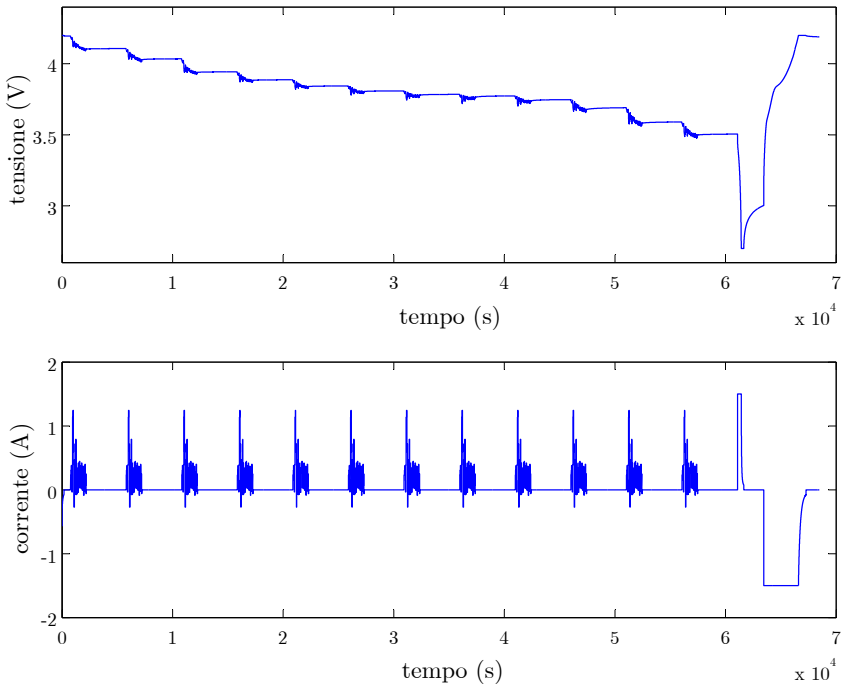

Figura 5.26: Tensione e corrente di test

Il risultato del test è riportato in Figura 5.27. Risulta evidente che le pause a corrente nulla consentono al sistema di compensare l'errore sulla stima, come ci si aspettava. Analizzando l'errore assoluto sulla stima del SoC (SoC_{err}) riportato in Figura 5.28, si nota infatti che esso si riduce durante le pause (che corrispondono ai tratti verticali, nei quali il SoC reale resta costante in quanto la corrente è nulla). Il RMSE e l'errore massimo in questo caso risultano pari a

- RMSE $= 3.11\%$
- $E_{max} = 9.35\%$

entrambi inferiori ai test precedenti.
Dall'esito di questo test è possibile inoltre osservare come varia il tempo di risposta del sistema in funzione del SoC. In Figura 5.27 si può infatti notare l'andamento simile ad un esponenziale della risposta del sistema durante le pause. Tale andamento, essendo appunto assimilabile ad un esponenziale, è caratterizzato da una costante tempo che dipende dal valore dello stato della

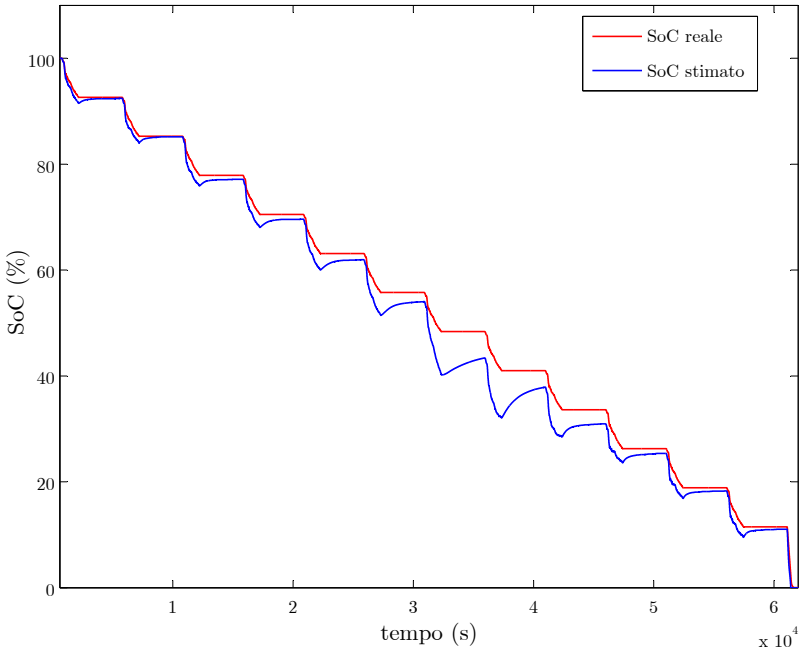

Figura 5.27: Esito del test

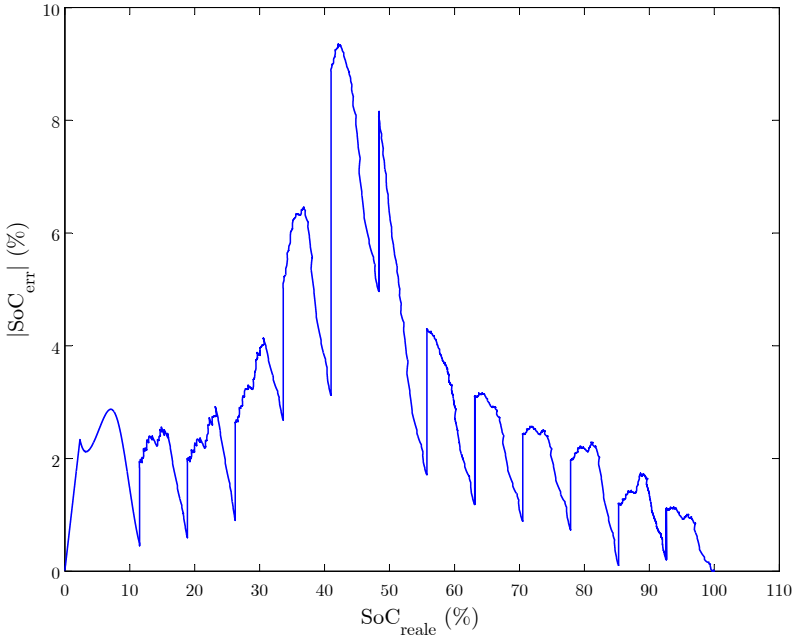

Figura 5.28: Errore assoluto sulla stima del SoC

cella, ovvero dal valore del SoC. Si nota che il valore delle costanti tempo risultano più elevati nella zona centrale per valori del SoC reale pari a circa il 40%-50% e decrescono per valori del SoC superiori o inferiori. In particolare nella zona indicata il SoC stimato non riesce nemmeno a raggiungere il valore di regime durante la pausa.
Queste osservazioni confermano dunque quanto affermato in precedenza: il sistema si comporta come un sistema di primo ordine caratterizzato da una costante tempo inversamente proporzionale alla pendenza della caratteristica SoC-OCV nel punto di lavoro.

Conclusioni

In questa tesi è stato presentato un algoritmo di stima dello stato di carica per batterie al litio-polimero con identificazione on-line dei parametri ed è stata svolta l'implementazione su FPGA dell'algoritmo di stima non comprensivo della parte di identificazione dei paramentri. Tale implementazione è stata svolta sulla scheda didattica Altera DE-0 (su cui è integrato l'FPGA Cyclone IV) e il sistema è stato sviluppato in ambiente Simulink tramite l'uso del tool di sviluppo DSP builder il quale ha consentito di effettuare una sintesi hardware automatica del modello generato e simulato in Simulink.
Per quanto riguarda la stima online dei parametri, è stato compiuto uno studio sulle tecniche più comuni di risoluzione dei problemi dei minimi quadrati e, per la risoluzione di tali problemi, sono stati proposti alcuni algoritmi adatti ad essere implementati efficacemente su FPGA.
Infine sono state svolte prove sperimentali riguardo il funzionamento del sistema su una cella di batteria NMC, allo scopo di verificare la validità sia dell'algoritmo di stima presentato, sia delle scelte implementative compiute. L'algoritmo di stima è stato dunque messo alla prova per un'applicazione reale, simulando le condizioni di funzionamento di un veicolo elettrico. A questo scopo sono stati effettuati tests utilizzando il profilo di velocità UDDS, ed il risultato della stima è stato confrontato con il valore di riferimento. L'esito di tali tests è stato soddisfacente, in quanto è risultato che il comportamento del sistema risulta coerente a quanto ottenuto a livello simulativo in Simulink. È stato inoltre valutato il comportamento del sistema in condizioni di normale funzionamento e in presenza di offset basandosi in particolare sull'errore risultante dal confronto della stima con il valore di riferimento.
Sebbene questo lavoro presenti un'implementazione ancora incompleta di un

algoritmo per la stima del SoC di una batteria, rispetto allo stato dell'arte questo lavoro fornisce buoni risultati, e ciò costituisce un invito a proseguire questo progetto per arrivare alla realizzazione di un completo sistema di stima del SoC. Gli argomenti trattati in questo lavoro di tesi possono infatti essere oggetto di ulteriore studio, in particolare per quanto riguarda la parte di identificazione on-line dei parametri. Dovrà dunque essere scelto un metodo di risoluzione del problema dei minimi quadrati, ad esempio uno tra quelli proposti in questa tesi, e dovrà poi essere implementato su FPGA come modulo hardware da far operare parallelamente al modulo di stima del SoC, così da completare l'implementazione dell'algoritmo di stima con identificazione on-line dei parametri.

Bibliografia

[1] A. Khaligh and Z. Li, "Battery, ultracapacitor, fuel cell, and hybrid energy storage systems for electric, hybrid electric, fuel cell, and plug-in hybrid electric vehicles: State of the art", *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, July 2010, pp. 2806-2814.

[2] J. Eyer and G. Corey, "Energy storage for the electricity grid: Benefits and market potential assessment guide. A study for the DOE energy storage systems program", *Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Rep. SAND2010-0815*, Feb. 2010.

[3] X. Chen, W. Shen, Thanh Tu Vo, Z. Cao and A. Kapoor, "An overview of lithium-ion batteries for electric vehicles", *IPEC, 2012 Conference on Power & Energy*, 2012, pp. 230-235.

[4] S. Piller, M. Perrin and A. Jossen "Methods for state-of-charge determination and their application", *Journal of Power Sources*, vol. 96, 2001, pp. 113-120.

[5] N. Watrin, B. Blunier and A. Mirauoi, "Review of adaptive systems for lithium batteries State-of-Charge and State-of-Health estimation", *Transportation Electrification Conference and Expo (ITEC)*, 2012, pp. 1-6.

[6] L. Lu, X. Han, J. Li, J. Hua, M. Ouyang, "A review on the key issues for lithium-ion battery management in electric vehicles", *Journal of Power Sources*, 2012, pp. 272-288.

[7] H. Rahimi-Eichi, U. Ojha, F. Baronti and M. Chow "Battery Management System, an overview of its application in the Smart Grid and Electric Vehicles", *IEEE Industrial Electronics Magazine*, June 2013, pp. 4-16.

[8] F. Baronti, G. Fantechi, R. Roncella, R. Saletti and P. Terreni, "Hardware buildings blocks of a hierarchical battery management system for a Fuel Cell HEV", *38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 4041-4047.

[9] M. Brandl, H. Gall, M. Wenger, V. Lorentz, M. Giegerich, F. Baronti, G. Fantechi, L. Fanucci, R. Roncella, R. Saletti, S. Saponara, A. Thaler, M. Cifrain and W. Prochazka, "Batteries and battery management systems for electric vehicles" *Proc. Design, Automation & Test in Europe Conference & Exibition*, 2012, pp. 971-976.

[10] J. Cao, N. Schofield and A. Emadi, "Battery balancing methods: a comprehensive review", *Vehicle Power and Propulsion Conference*, 2008, pp. 1-6.

[11] F. Baronti, W. Zamboni, N. Femia, H. Rahimi-Eichi, R. Roncella, S. Rosi, R. Saletti and M.-Y. Chow, "Parameter Identification of Li-Po Batteries in Electric Vehicles: a Comparative Study", *Industrial Electronics (ISIE), 2013 IEEE International Symposium*, 2013, pp. 1-7.

[12] M. Charkhgard, M. Farrokhi, "State-of-Charge Estimation for Lithium-Ion Batteries Using Neural Networks and EKF" *Industrial Electronics, IEEE Transactions*, Dec. 2010, pp. 4178-4187.

[13] C. Unterrieder, R. Priewasser, M. Agostinelli, S. Marsili and M. Huemer, "Comparative study and improvement of battery open-circuit voltage estimation methods" *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium*, Aug. 2012, pp. 1076-1079.

[14] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 1. Background", *Journal of Power Sources 134*, 2004, pp. 252-261.

[15] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 2. Modeling and identification", *Journal of Power Sources* vol. 134, 2004, pp. 262-276.

[16] S. Rosi, "Stima dello stato di carica con identificazione on-line dei parametri di batterie al litio-polimero", *Tesi di Laurea Specialistica, Università di Pisa*, Dec. 2012.

[17] M. Chen and G. Rincon-Mora, "Accurate Electrical Battery Model Capable of Predicting Runtime and I-V Performance", *IEEE Transaction on Energy Conversion*, vol.21, no.2, Jun. 2006, pp. 504-511.

[18] H. Rahimi-Eichi, F. Baronti and M.-Y. Chow, "Online Adaptive Parameter Identification and State-of-Charge Coestimation for Lithium-Polymer Battery Cells", *Industrial Electronics, IEEE Transactions*, vol. 61, n. 4, Apr. 2014, pp. 2053-2061.

[19] I. Snihir, W. Rey, E. Verbitskiy, A. Belfadhel-Ayeb and P.H.L. Notten "Battery open-circuit voltage estimation by a method of statistical analysis", *Journal of Power Sources*, vol. 159, 2006, pp. 1484-1487.

[20] M. Dubarry, V. Svoboda, R. Hwu and B.Y. Liaw "Capacity loss in rechargeable lithium cells during cycle life testing: The importance of determining state-of-charge", *Journal of Power Sources* vol. 174, 2007, pp. 1121-1125.

[21] H. Rahimi-Eichi, F. Baronti and M.-Y. Chow, "Modeling and online parameter identification of Li-Polymer battery cells for SoC estimation", *2012 IEEE International Symposium on Industrial Electronics*, May 2012, pp. 1336-1341.

[22] H. Rahimi-Eichi and M.-Y. Chow, "Adaptive parameter identification and State-of-Charge estimation of lithium-ion batteries", *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronic Society*, Oct. 2012, pp. 4012-4017.

[23] staff of Berkeley Design Technology, Inc., “Floating-point DSP Design Flow and Performance an Altera 28-nm FPGAs”, Oct. 2012, pp. 1-14.

[24] O. Maslennikow, V. Lepekha, A. Sergiyenko, A. Tomas and R. Wyrzrkowski, “Parallel implementation of Cholesky LLT-Algorithm in FPGA-Based Processors”, *Parallel Processing and Applied Mathematics*, 2008, pp. 137-147.

[25] G. Golub, C. Van Loan, “Matrix Computations” *The Johns Hopkins University Press*, 4th edition, 2013.

[26] P. Luethiy, C. Studery, S. Duetschz, E. Zgraggenz, H. Kaesliny, N. Felbery, and W. Fichtner, “Gram-Schmidt-based QR Decomposition for MIMO Detection: VLSI Implementation and Comparison”, *IEEE Asia Pacific Conference* Dec. 2008, pp. 830-833.

[27] C. Singh, S. Prasad and P. Balsara “VLSI Architecture for Matrix Inversion using Modified Gram-Schmidt based QR Decomposition” *20th International Conference on VLSI Design*, Jan. 2007, pp. 836-841.

[28] D. Chen and M. Sima, “Fixed-Point CORDIC-Based QR Decomposition by Givens Rotations on FPGA”, *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec. 2011, pp. 327-332.

[29] S. Wang and E. Swartzlander, “The Critically Damped CORDIC Algorithm for QR Decomposition”, *Conference Record of the Thirtieth Asilomar Conference on Signals, Systems and Computers*, Nov. 1996, vol. 2, pp. 908-911.

[30] S. Aslan, S. Niu and J. Saniie, “FPGA Implementation of Fast QR Decomposition based on Givens Rotation”, *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2012, pp. 470-473.

[31] D. Yang, G. Peterson, H. Li and J. Sun “An FPGA Implementation for Solving Least Square Problem”, *17th IEEE Symposium on Field Programmable Custom Computing Machines*, Apr. 2009, pp. 303-306.

[32] A. Rahmoun and H. Biechl, "Modelling of Li-ion batteries using equivalent circuit diagrams", *PRZEGLĄD ELEKTROTECHNICZNY (Electrical Review)*, 2012, pp. 152-156.

[33] "Urban Dynamometer Driving Schedule (UDDS)" [Online]. Available: http://www.epa.gov/nvfel/testing/dynamometer.htm.